

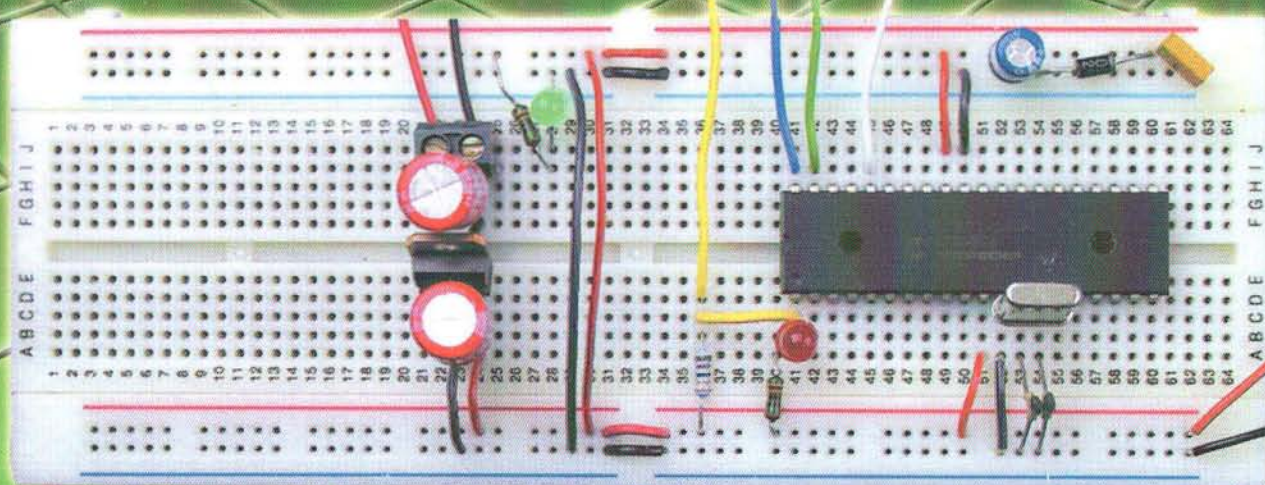
MacTech[®]

The Journal of Macintosh Technology and Development

PIC Microcontroller Programming on Mac OS X

By Tom Djajadiningrat

*Using a VOTI Wisp628
with JAL and XWisp*



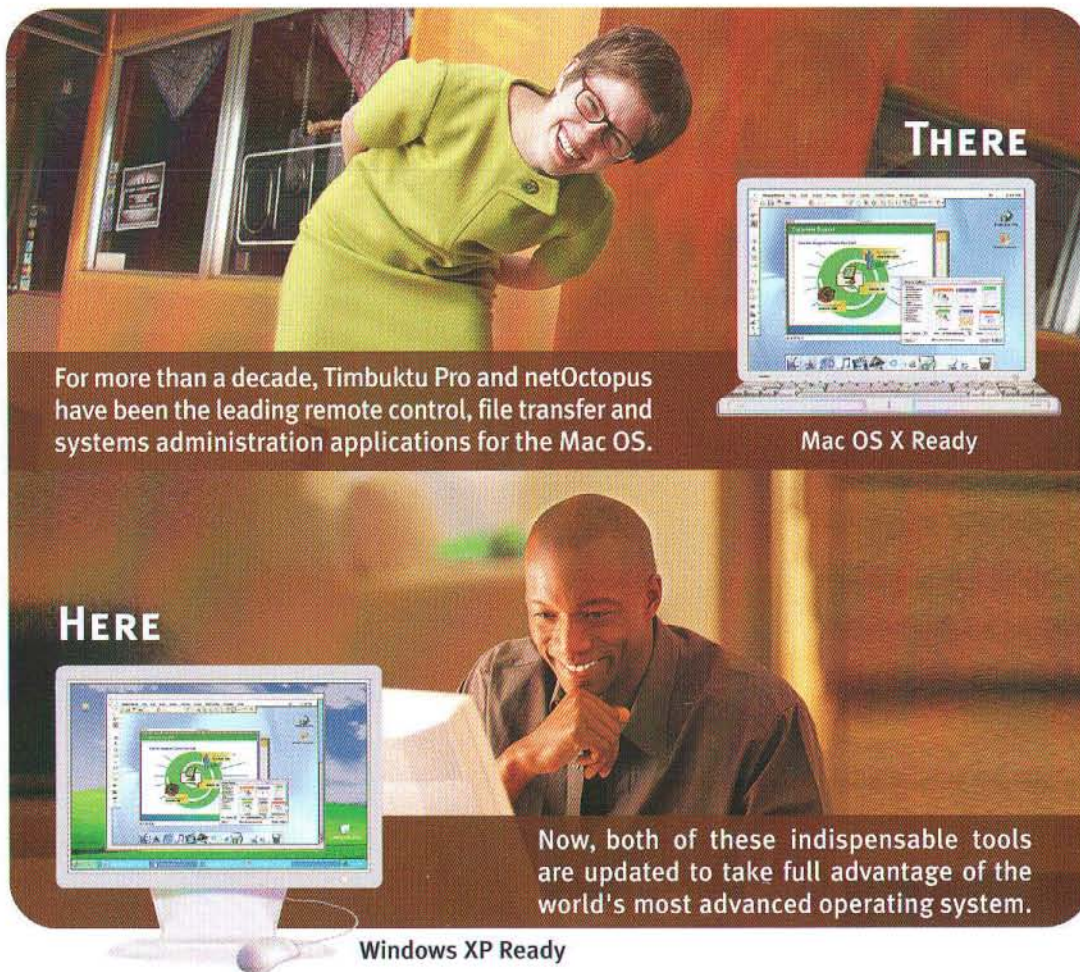
05 US
95 Canada
N 1067-8360
nted in U.S.A.



02

3212874887 8

Stay In Control Wherever You Go.



THERE

For more than a decade, Timbuktu Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS.

Mac OS X Ready

HERE

Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Windows XP Ready

Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

<http://www.timbuktopro.com>

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.

<http://www.netoctopus.com>

Learn more, try it, or buy it online. Call us at **1-800-485-5741**.



timbuktu® • netOctopus®

netopia®

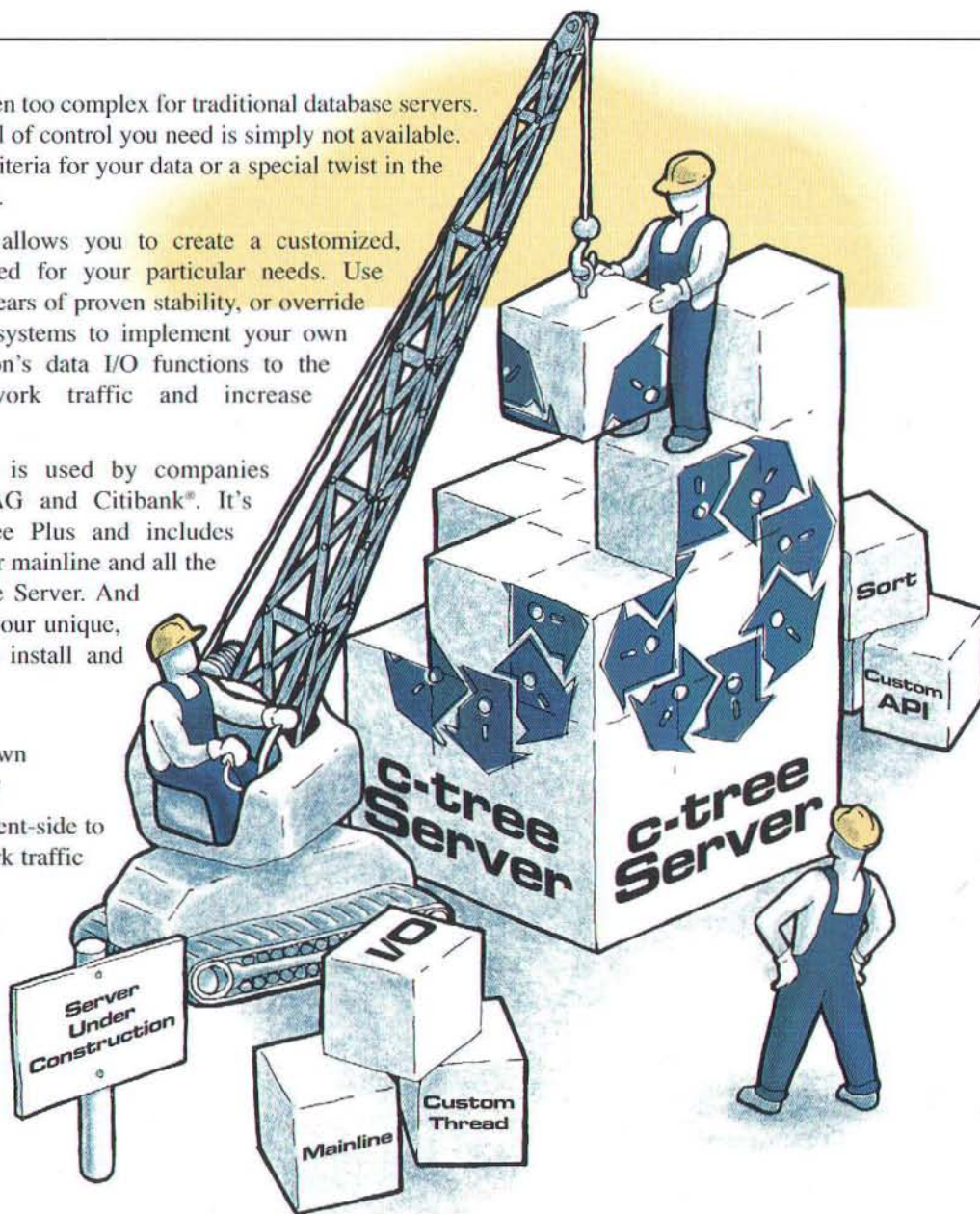
CUSTOMIZE YOUR DATABASE SERVER WITH THE C-TREE® SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree® Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank®. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!



FairCom®
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Editor-in-Chief • Dave Mark

Managing Editor • Jessica Stubblefield

Regular Columnists

Getting Started

by Dave Mark

QuickTime Toolkit

by Tim Monroe

Mac OS X Programming Secrets

by Scott Knaster

Reviews/KoolTools

by Michael R. Harvey

Patch Panel

by John C. Welch

Section 7

by Rich Morin

Untangling the Web

by Kevin Hemenway

John and Pals' Puzzle Page

by John Vink

Regular Contributors

Vicki Brown, Erick Tejkowski,
Paul E. Seving

MacTech's Board of Advisors

Jordan Dea-Mattson, Jim Straus
and Jon Wiederspan

MacTech's Contributing Editors

- Michael Brian Bentley
- Vicki Brown
- Marshall Clow
- John C. Daub
- Tom Djajadiningrat
- Bill Doerfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Sun
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Rich Morin
- John O'Fallon, Maxum Development
- Will Porter
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Chuck Von Rospach, Plaidworks

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Account Executive • Lorin Rivers
adsales@mactech.com • 800-5-MACDEV

Events Manager • Susan M. Worley
International • Rose Kemps

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane, Susan Pomrantz

Shipping/Receiving • Dennis Bower

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2003 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-I, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

February 2004 • Volume 20, Issue 2

COVER STORY

32 PIC Microcontroller Programming on MacOSX

Using a VOTI Wisp628 with JAL and XWisp

By Tom Djajadiningrat, Designed Intelligence Group, Eindhoven University of Technology

MWSF '04

48 Round-up of what we saw this year

By Michael R. Harvey

QUICKTIME TOOLKIT

58 Swing Shift

Editing QuickTime Movies with Java

By Tim Monroe

CASTING YOUR .NET

20 The SSCLI: Managed Languages, Samples and (more) Programming Utilities

Exploring .NET development on Mac OS X

By Andrew Troelsen, Minneapolis, Minnesota

GETTING STARTED

4 More Fun With AppleScript

By Dave Mark, Editor In Chief

RAPID DEVELOPMENT

42 NoCode Browser

Using the Apple's Web Kit SDK to make a web browser

By David Linker

UNTANGLING THE WEB

12 Your First MySQL Data Entry

Finally, let's define your database tables and enter some data.

By Kevin Hemenway, Masticating Orifice

MAC OS X PROGRAMMING SECRETS

8 Ship It! Distributing Your Software, Part 2

By Scott Knaster

SOFTWARE MARKETING

52 Sweetening the Deal

Increasing Software Sales with Purchase Incentives

By Dave Wooldridge

STARTING A BUSINESS

16 Life Change Ahead

Starting a business will change your life. How it will change is up to you.

By Chris Kilbourn

REVIEWS

68 REALbasic Review

A better Visual Basic than Visual Basic

By Guyren G. Howe

Copyright 2003 by Dave Mark

Getting Started: More Fun With AppleScript

In last month's column, we installed Script Menu to help us organize our scripts, and then started digging into the Finder's AppleScript dictionary. Fantastic stuff! Hopefully, you've had a chance to play. If not, launch Script Editor, select *Library* from the *Window* menu, then double-click on the *Finder* entry in the *Library* list. A Finder dictionary window should appear.

Click on the line in the left pane labeled *Finder Basics*. The classes and commands associated with the *Finder Basics* AppleScript suite will appear (see **Figure 1**). Notice that the *Finder Basics* are divided into *Properties* and *Elements*. Think of properties as class data and the elements as objects within the class.



Figure 1. The Finder Basics dictionary entries.

For example, go into the Script Editor and type this script:

```
tell application "Finder"
    get name
end tell
```

When you run this script, here's what appears in the results field:

"Finder"

This is the value of the *name* property of the Finder. If you are running Panther, here's the result of the command *get version*:

"10.3"

Here's an interesting one. First, do a *get startup disk*:

```
startup disk of application "Finder"
```

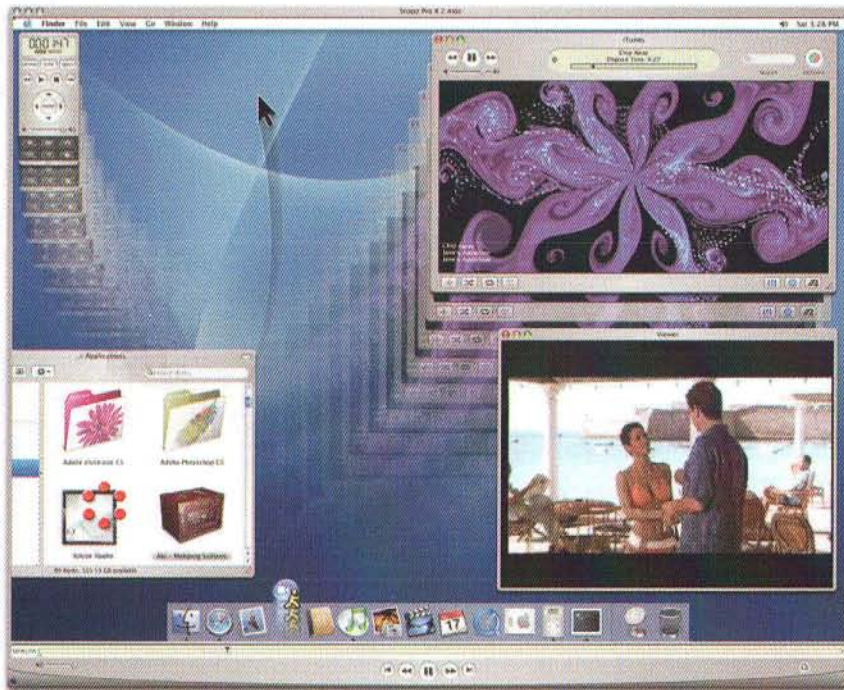
Now do a *get name of startup disk*:

"Macintosh HD"

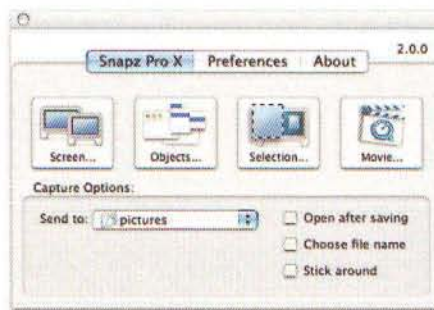
See the difference? *Startup disk* is a reference to the startup disk, whereas *name of startup disk* is a string containing the name of the startup disk. If you look in Figure 1, you'll see that *startup disk* is of type *disk*. The *disk* class is defined in the *Containers and folders* suite. The class is shown in Figure 2. Notice that the dictionary shows the plural form of disk as *disks*. The dictionary also shows the elements and properties of the disk class.

Dave Mark is a long-time Mac developer and author and has written a number of books on Macintosh development, including *Learn C on the Macintosh*, *Learn C++ on the Macintosh*, and *The Macintosh Programming Primer* series. Be sure to check out Dave's web site at <http://www.spiderworks.com>.

If a picture is worth a thousand words,
imagine how priceless a movie would be...



Snapz Pro X 2.0 allows you to effortlessly record anything on your screen, saving it as a QuickTime® movie that can be emailed, put up on the web, or passed around however you please.



Why take a static screenshot when Snapz Pro X 2.0 makes creating a movie just as easy?
Snapz Pro X 2.0 does that, and so much more -- what a difference a version makes!
Download a free demo version from our web site today and see for yourself:

<http://www.AmbrosiaSW.com/>



Snapz Pro X 2.0
AMBROSIA®
SOFTWARE INC



Snapz Pro X

Snapz Pro X, Ambrosia Software, Inc., and the Ambrosia Software logo are registered trademarks of Ambrosia Software, Inc.

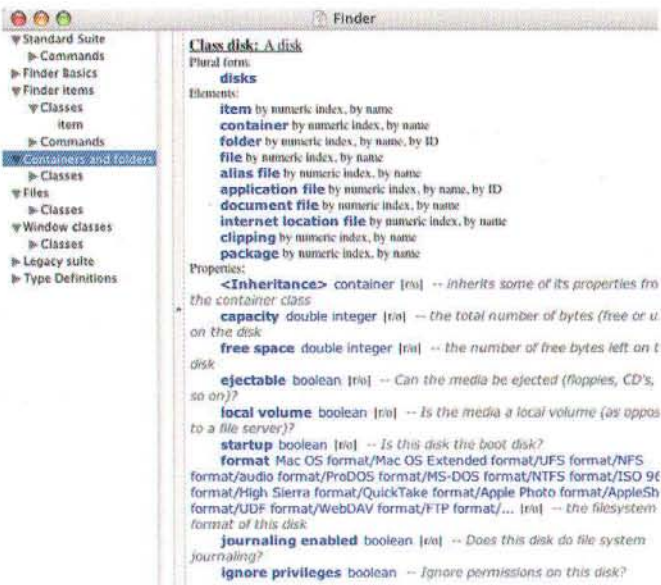


Figure 2. The disk class entries of the Containers and folders suite.

Try the command `get format of startup disk`. Remember, `format` is a property of the `disk` class and `startup disk` is defined as being of class `disk`. Here's the result on my Mac:

```
Mac OS Extended format
```

Now try `get journaling enabled of startup disk`. Here's the result:

```
true
```

As you might expect, Boolean properties return either true or false. The command `get ignore privileges of startup disk` returns:

```
false
```

Finder Basics - Elements

Go back to Figure 1 and take a look at the elements in the Finder Basics suite. These are the objects that you might expect the Finder to encounter. What do you think the result will be of running this script:

```
tell application "Finder"
  get items
end tell
```

It helps to have a sense of the `item` class. An item is basically a generic object – pretty much anything that could exist in a Finder window or on the desktop. Here's the result I got when I ran the script on my Mac:

```
{startup disk of application "Finder", disk "Macintosh HD" of application "Finder", folder "AE Monitor 1.0" of desktop of application "Finder", document file "DSC_0751.JPG" of desktop of application "Finder", folder "InDesign CS Example Scripts"
```

of desktop of application "Finder", folder "UI Element Inspector" of desktop of application "Finder", alias file "Warcraft III" of desktop of application "Finder", folder "wedding_pics" of desktop of application "Finder", document file "Xcode notes" of desktop of application "Finder"]

Look through this list of items. Note the curly braces, necessary to making this a proper list. Take a look at the nomenclature, especially the use of the word "of" all over the place. Think containers/enclosures, items within items within items.

Figure 2 shows the listing of the Finder items suite in a dictionary window. Check out the list of properties. Note that an item has an index. Look in the list of items in the previous list, returned by the `get items` command. What do you think will happen when I run the command `get item 2`. Here's my result:

disk "Macintosh HD" of application "Finder"



Figure 3. The item class in the Finder items suite.

As you might expect, `get item 2` returns the second item in the list returned by `get items`. Now, what do you suppose you'd get from this command:

```
get container of item 2
```

In my case, I got:

```
folder "Dave Mark's Computer" of application "Finder"
```

Brave New Make

So far, all we've done is suck information out of the Finder. Let's try making something.

In the Finder's dictionary window, click on the *Commands* line in the Standard Suite. The commands supported by all subsequent suites are listed. One of these is the *make* command. The details are shown in **Figure 4**.



Figure 4. Details on the *make* command.

Here's a script to create a new folder at the current insertion location (the desktop).

```
tell application "Finder"
    make new folder with properties {name:"blap"}
end tell
```

This tells the Finder to create a new folder at the current insertion location (since I didn't specify a new location) with the specified properties. Since I only specified a single property, this will be pretty simple. I ran this script and a new folder named "blap" appeared on my desktop. Here's the result returned by my script:

```
folder "blap" of folder "Desktop" of folder "davemark" of
folder "Users" of startup disk of application "Finder"
```

Does this make sense? You can see the containment hierarchy as it walks its way out from the new folder to the top of my hard drive and Finder itself.

Now that "blap" exists, let's try running the script again. Hrm. As you can see in **Figure 5**, we got an error, telling us that the folder already exists.

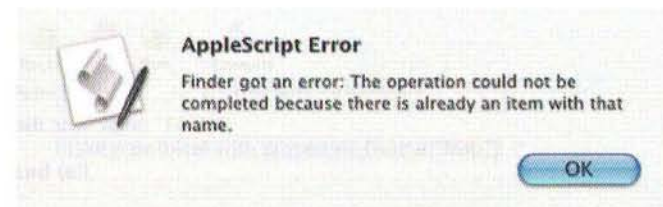


Figure 5. An error is reported when I try creating a folder that already exists. Hrm.

Having Script Editor report the error is very useful if you are testing your script or just playing. But if you deliver this script as a compiled entity to a user, you'll want the program to handle the errors itself. Here's a new version of the script:

```
tell application "Finder"
```

```
    try
        make new folder with properties {name:"blap"}
    on error error_message
        display dialog error_message
    end try
end tell
```

Running this script produces the dialog shown in **Figure 6**. Much better. Now the script uses a *try* block and a corresponding *on error* block to pass the incoming error message to the *display dialog* command.

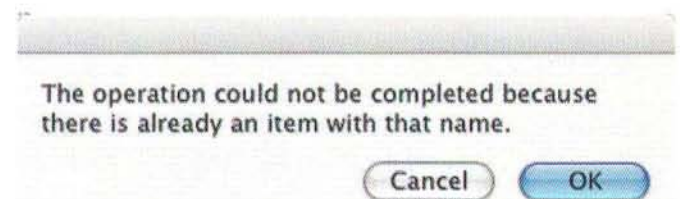


Figure 6. A simple error dialog.

Here's the result of running this script (I clicked the OK button):

```
[button returned:"OK"]
```

TILL NEXT MONTH...

Hopefully, you are getting a bit more familiar with the AppleScript Finder dictionary and the way you can take advantage of properties, elements, and commands. Now that you have a sense of the Finder, open up some other dictionaries. A great one to play with is the TextEdit dictionary. A lovely place to create text elements and copy them to the clipboard for use in other applications. Try your hand at manipulating TextEdit. See what you can do with it.

We *will* return to AppleScript at some point (I love it too much not to) but next month's column is going back to Cocoa. I am heading down to Atlanta to Aaron Hillegass' Big Nerd Ranch for a week of intensive Cocoa training. I am *very* excited about this. Model View Controller, Model View Controller. See you next month! ☺



GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See **www.lemkesoft.com** for more information.

By Scott Knaster

Ship It!

Distributing Your Software, Part 2

In our last thrilling installment (or was it a thrilling *installation*?), we discussed the simplest method of distribution for your Mac OS X software: disk images. We talked about using Disk Copy or Disk Utility to make a disk image, and we described the relatively recent technology of Internet-enabling a disk image so that it cleans up after itself when users download it in Safari.

When you distribute software on a disk image, there is no actual installing taking place. Users simply drag and drop files onto their hard disks. That's the great strength and weakness of disk images: they're easy for you to make and easy for users to figure out, but you can quickly run into their limitations. For example, what if you need to make sure files end up in specific places on the disk? What about checking the progress of the installation and taking some action based on what's happening? Most commercial applications need to do things like this when they're installed. For more control of the installation process, Apple provides the PackageMaker and Installer tools.

MEET YOUR MAKER

PackageMaker is an application that comes with Mac OS X. You use PackageMaker to create packages, which are documents with extension .pkg that contain files to be installed and related files used during the installation. An Apple application called Installer reads the information from a package and performs the installation.

The two biggest tricks PackageMaker installations can do that disk images can't are installing files anywhere in the system, and running scripts while the installation is happening. To install files where you want them, you'll create a directory structure in your package that shows how you want files installed in the system, and includes a few extra items used by the installation itself. In order to run scripts, Installer defines certain events in the installation process and looks for scripts with names corresponding to those events. If the scripts exist, Installer runs them.

Let's take a deeper look at setting up your files and scripts for installation. Please gaze directly into the magazine.

EVERYTHING IN ITS RIGHT PLACE

You'll make a set of directories and files that indicates where you want everything to end up after the installation. The basic pieces of the directory structure are as follows:

- You'll create a directory that contains all the other directories and files in the package. This directory can have any name, such as "PackageContents" or "Fred". When you run PackageMaker, you locate this folder.
- Within the all-enclosing directory, you'll have a directory that holds various files used for installation, such as install-time scripts, the license and readme docs, and so on. Again, this directory can have any name – you specify it in PackageMaker.
- A second directory inside the outermost directory holds the files to be installed. They're arranged in directories according to the way you want them to be installed. When you run PackageMaker, you specify whether you the files can be installed anywhere or must be placed in absolute locations in the system.

Here's an example. Let's say we're going to install a package that consists of a music application, 3 system sounds, and a screen saver module. Our package directory structure would look like this:

```
PackageContents
PackageRoot
  Applications
    Music2MyEars
  System
    Library
      Sounds
        sound1.aiff
        sound2.aiff
        sound3.aiff
      Screen Savers
        ears.saver
  PackageResources
```

Scott Knaster has been playing with Apple computers, and occasionally working with them, since 1977. Wow, that's a long time! Scott has written books and articles for normal human beings as well as programmers. Photo of Scott and his friend Jollibee courtesy of John Vink Photography.

IN THE SOPHOS ZONE, VIRUSES NEVER INTERRUPT YOUR WORKFLOW



Sophos Anti-Virus provides total protection against all known viruses on Mac OS X. Its GUI enables users to perform an immediate scan of any accessible file, folder and volume, while InterCheck technology keeps users safe by intercepting all file accesses and scanning for viruses in the background in real-time. In addition, every license includes 24x7x365 technical support. In the Sophos Zone, viruses don't stand a chance.

For a free, fully supported evaluation, visit:
www.sophos.com/av_mac

www.sophos.com/av_mac
Tel 888-216-6703

SOPHOS
engineered for business


```
License.rtf
ReadMe.rtf
preflight
preinstall
postinstall
postflight
```

The files in the PackageRoot directory are the software itself. We want the files installed in specific locations. When we run PackageMaker to create the package, we can make sure that will happen.

The PackageResources directory is filled with files that will be used during the installation. These include the ReadMe and license documents, along with several scripts (preflight, preinstall, etc.) that Installer will run. In this example, our text files are RTF, but you can also use .txt or .html files.

STAYING IN CONTROL

When the user runs Installer, usually by double-clicking a .pkg file, these are the steps that take place:

- Installer looks in the package for an InstallationCheck script and runs it, if it's there. This script provides an opportunity to cancel the installation by returning a nonzero value.
- If there's a Welcome file (such as Welcome.rtf or Welcome.html), Installer displays it. Otherwise, a default Welcome screen appears. If there's a background image (a file named "background" with extension .tiff, .jpg, .gif, .pdf, or .eps), it's displayed behind the Welcome.
- If there's a ReadMe file (such as ReadMe.rtf or ReadMe.html), it's displayed next.
- Next, the License file is displayed.
- When the user gets to the Select a Destination screen, Installer runs the VolumeCheck script (if it exists) for each mounted volume. VolumeCheck can return a nonzero value to prevent a volume from being eligible for installation.
- After the user clicks Install, the pflight script is run. Like the other scripts, pflight can cancel the installation by returning a nonzero value.
- If the user is upgrading the package from a previous version, Installer runs the preupgrade script. Otherwise, preinstall is run.
- The package contents are copied to the destination. Actual installing, wow!
- If this was an upgrade, the postupgrade script is run. Otherwise, postinstall is run. There's still time to cancel the installation, which happens if these scripts return nonzero. Installer runs the postflight script, if it's present.

As you can see from these steps, you have a chance to check the installation with many different scripts: InstallationCheck and VolumeCheck, pflight, preinstall or preupgrade, postinstall or postupgrade, and postflight. You should specify a shell in the first line of your script, using the `#!/bin/sh` convention. Installer provides a bunch of arguments and environment variables you can use in the scripts. For example, the first argument (\$1) is the full path

to the installation package, \$2 is the path to the destination, and \$3 is the mountpoint of the destination volume.

PACK IT UP

Once you have written the scripts you want, created the ReadMe and related files, and assembled everything in the right directories with your installation files, it's time to make the package itself. You do this by running PackageMaker, which is located in Developer/Applications/Utilities/. When you run PackageMaker, you see the lovely screen that appears in **Figure 1**. It's safe to say Jonathan Ive hasn't gotten hold of this one yet.



Figure 1. This is the screen that appears when you launch PackageMaker. It's not very pretty, but it works just fine.

To start assembling your package, fill out the fields on the Description tab. The Title and Description fields provide a place for information that's visible to users when the Installer is running. Click the Files tab to specify the source location of the files to be installed. In our example above, this is the PackageRoot directory. Just to be macho, note that the field for the directory is editable text, so if you're really geeky and you hate clicking through a file dialog, you can actually type the directory path yourself.

Next, use the Resources tab to specify where the scripts, ReadMe, and similar files are located. In our earlier example, this was the PackageResources directory.

The Info tab provides a few interesting options. For example, you can choose whether to recommend or require a restart here. Restarting is mean and should be avoided whenever possible. (In Panther, you can also specify that logout is required, but you can't do it here: you have to modify the Info.plist to do so. To make this happen, add an "IFPkgFlagRestartAction" key with a value of "RequiredLogout" to the package's Info.plist.) If your software doesn't have to be installed in a particular location, check the Relocatable box and users will get to pick where your software should go. The last tab, Version, lets you specify info that appears in the Finder.

When you have your package just the way you want, you should save in PackageMaker, then choose File → Create Package to make the package. When you double-click the package, Installer will launch and you'll get the exciting display you see in **Figure 2**.



Figure 2. The Installer opening screen can display custom text and background image.

There are a few more Installer features that aren't available in PackageMaker. Instead, you have to edit the package's Info.plist file to get them. For example, when you include a background image in your package, Installer stretches it out to fill the entire window, which is ugly if the image has the wrong proportions. You can gain more control over how the image is used by editing the Info.plist file with Property List Editor.

In the package shown in **Figure 2**, the Info.plist file has two new keys added to it: IFPkgFlagBackgroundScaling and IFPkgFlagBackgroundAlignment, to make sure that that bee and that guy don't take over the screen. The first key, which controls how the image is scaled, is set to "none". You can also use "tofit" (the default) and "proportional". The second key, for alignment, is set to "bottomleft". Other valid alignments are "left", "right", "top", "bottom", "center" (the default), "topleft", "topright", and "bottomright".

PackageMaker has a few more tricks, such as the ability to create *metapackages*, which are packages that contain and install other packages. Metapackages can present and sort through complex sets of dependencies for installing software. Another cool feature is the ability to install packages across a network, which you can do if you have Apple Remote Desktop.

INSTALLERMANIA

PackageMaker isn't the last word in installers. Lots of commercial software requires more control than you can get from PackageMaker and Installer. The best-known non-Apple installer packages are made by MindVision Software (InstallerVise, ViseX, and FileStorm), and Aladdin (StuffIt

InstallerMaker). These applications let you do things that are beyond PackageMaker, such as creating installs that work on OS 9 or Windows, displaying multiple "tips" screens during install, and automatically adding items to the OS X dock. These products compete with free software from Apple, so they have to be feature-packed.

You can find out more, or even download whole manuals and applications, by visiting the web sites at www.mindvision.com and www.stuffit.com/mac/installermaker/. MindVision downloads are fully functional, with limits on the installers you build until you buy a registration key. The InstallerMaker download is fully functional, but produces installers that say you're using an unregistered copy.

Speaking of free stuff, Apple provides a pile of fine documentation on packages, PackageMaker, and related topics. For much more detail on the stuff we talked about in this article, check out the following URL:

<http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution>.

Have fun getting your installer together. Remember, no matter how hard you work on your application, nobody will be able to use it unless you produce an installer, disk image, or other trick to get it out into the world.

TelnetLauncher

Bookmark and Launch your
Telnet and SSH sessions
Featuring:

- >Password storage
- >Auto minimize
- >Set text font and size
- >Specify terminal colors
- >Port forwarding
- >Terminal emulation

This and much more available from...

piDog Software

SimpleKeys - piPop - DockSwap - ScreenShot Plus

www.pidog.com Info@pidog.com

By Kevin Hemenway, Masticating Orifice

Your First MySQL Data Entry

Finally, let's define your database tables and enter some data.

In our past few issues, we've explored installing MySQL and creating a database and user with `mysql_setpermission`. Along with their access permissions and capabilities, we've verified that things had gone smoothly by using `/Library/MySQL/bin/mysqlshow -u root -p` to show the list of MySQL databases currently available (in which `mactech`, our in-progress database, was visible). What we've yet to do is actually define our database (what sort of data it'll be representing) or insert any of the data itself.

Before we do, allow me to demonstrate another timesaving tip: `.my.cnf`.

SAVING KEYSTROKES WITH YOUR PERSONAL .MY.CNF

As we've been issuing various MySQL commands, we've always had to be careful to pass MySQL username and password – certifying to MySQL that even though we're the OS X `morbus` user, we really want to act as the MySQL `root` user. Depending on how confident you are with your computer's security, you can remove the need for the username and password for most future MySQL related commands.

To do so, open up a blank text document and add the following lines:

```
[client]
user=root
password=yourMySQLrootpassword

# The following configuration definition is optional.
# but becomes useful if you were connecting to a MySQL
# server on an entirely different machine. We'll
# leave it commented here, since we'll only be
# accessing the local MySQL installation.
#
# hostname=some.other.server.com
```

You'll want to save this file as `.my.cnf` in your home directory (`/Users/morbus/.my.cnf`, in my case). Be sure to include the beginning dot: that's required for the configuration to be read properly, but it will also (depending on your settings) hide it from view of the OS X Finder and prying, but unskilled, eyes.

Since this new file contains the MySQL root password in plain text, we want to ensure that only our account can read the file. To do this from the Terminal, run the following command, which gives read and write permissions only to the `morbus` user: `chmod 600 ~/.my.cnf`. Alternatively, if you can see the file in the Finder, "Get Info" on it, and expand "Owners & Permissions", then "Details". Give the "Owner" "Read & Write" access, and the "Group" and "Others" should have none. Your final results should look like **Figure 1**.



Figure 1: The access permissions of the `.my.cnf` file.

With the `.my.cnf` file in place, you should now be able to run `/Library/MySQL/bin/mysqlshow` without requiring a username and password. Likewise, some other MySQL utilities can have other options preset to save keystrokes. For instance, `mysqlshow test` will show you the defined tables of the `test` database:

```
~ > mysqlshow test
Database: test
+-----+
| Tables |
+-----+
| testac |
| testad |
| testae |
| testaf |
+-----+
```

But adding the `—status` flag will give you a much healthier dose of information (which is so wide we can't easily replicate it here). If you find yourself issuing the `—status` flag a lot, you can save yourself time by adding that to `.my.cnf` under an application-specific block:

```
[mysqlshow]
status
```

Kevin Hemenway, coauthor of *Mac OS X Hacks* and *Spidering Hacks*, is better known as Morbus Iff, the creator of `disobey.com`, which bills itself as "content for the discontented." Publisher and developer of more home cooking than you could ever imagine (like the popular open-sourced aggregator `AmphetaDesk`, the best-kept gaming secret `Gamegrene.com`, the ever ignorable `Nonsense Network`, etc.), he has started his indexing and cataloging project two months earlier than he intended. Contact him at `morbus@disobey.com`.

The basic rule is: if the MySQL utility accepts a double-dash command line flag (like `--status`, `--username`, `--password`, etc.), then you can define it as a preset within `.my.cnf` under a block named after the application. This can occasionally be useful, but I find myself using shell aliases far more frequently (which are outside the jurisdiction of this column.)

DEFINING THE MACTECH DATABASE

Anytime you interact with a MySQL database, you'll be talking to it in a language called SQL, or "Structured Query Language". SQL is an industry standard for database communication, and all known databases support it in some way (and occasionally, differently than what the standard suggests). We won't be teaching you a lot of SQL in these columns, instead focusing on just enough to get you started. The complete reference of the SQL implementation of MySQL can be found at <http://www.mysql.com/>.

The first bit of SQL you'll learn is how to define the database you're creating. When these SQL definitions are included along with other documentation on how you've approached the database design, you've created a "database schema". Schema's can be immensely helpful as a reference while your writing code, as well as when you later need to define a new query.

Take a look at **Listing 1**, which contains the SQL statements to define our `mactech` database. Save this into `mactech.sql`—we'll be piping it to a command line utility shortly.

Listing 1: Our database definition, in SQL.

`mactech.sql`

```
CREATE TABLE person (
  id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  name VARCHAR(255),
  date_of_birth DATE,
  date_of_death DATE,
  title VARCHAR(50),
  designation VARCHAR(255)
```

```
);

CREATE TABLE books (
  id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
  title TINYTEXT,
  publication DATE,
);

CREATE TABLE relationships (
  person_id INT NOT NULL,
  book_id INT NOT NULL,
  INDEX (person_id,book_id)
);
```

There are three SQL statements in **Listing 1**, and the semi-colon that completes them is required for all statements you write. Each individual statement creates one table, being a set of columns that describe your data, and the rows that contain it. If you know HTML, they're exactly the same as the `<table>` tag (assuming it's not used for layout).

The first table, `person`, contains fields (or columns) that define an individual: their name, date of birth and possible death, title (which could be ranks like "Major", "President", or addresses like "Sir", "Ms", etc.) and designation (which could be succession like "Jr.", "III", "the Brave", etc.). The second table gives a rather meager definition for `books`, containing only their title and publication date.

You'll notice that each column is also described by what sort of data they'll contain. A person's `name` can be a variable length, with a maximum of 255 characters, whereas strings like `date_of_birth`, `date_of_death`, and `publication` are `DATES`, funnily enough. A person's `title` has been limited to a maximum of 50 characters, whereas `designations` could be much longer (assuming extenuating circumstances like "the Brave Sir Knight of Camelot who has Sleweth the Dragon of Ill Contempt!") There are over a dozen different types of database columns; definitions of each can be found in the MySQL web documentation.

Our `person` and `books` tables each contain a column called `id`, which auto increments by 1 for every new row of data



Now serving Cocoa[®] just the way you want it.

Training for Mac OS X doesn't have to be the same old flavor. Reserve your seat in a class at our scenic lodge location, or have experts come to you for **Extreme Mentoring**. Two weeks of on-site instruction and collaboration, customized to the requirements of your project. Book now for 2003. See why we're different.



**Big
nerd
ranch**

Intensive Classes for Programmers
www.bignerdranch.com

entered into that table. These serve to uniquely identify that particular record—person #15 today is going to be the same person #15 three weeks from now. As such, these columns must always have an integer value, as defined by INT and NOT NULL. Since AUTO_INCREMENT is set, MySQL will happily fill this data in for us—we'll only need to worry about it when we're selecting that record ("gimme person #134!"). A PRIMARY KEY, which is similar to the INDEX in our so-far-ignored third table, will make these lookups of data more efficient.

Our final table, *relationships*, contains two columns that represent ids from our other two. This is what makes a "relational database" relational: they're designed to relate records in one table with records in another, creating a communal result. Entire books have been written on how to smartly design databases, but let's go over a quick example to show how a relational database can ensure data integrity. Say you've the database in Listing 2, with a sampling of the data within shown in Listing 3.

Listing 2: A non-relational database.

non-relational.sql

```
CREATE TABLE works (
  person_name VARCHAR(255),
  book_title TINYTEXT
);
```

Listing 3: Some sample data from our non-relational database.

non-relational.txt

person_name	book_title
Kevin Hemenway	Mac OS X Hacks
Kevin Hemenway	Spidering Hacks
Kevin Hemenway	Advanced Procrastination
Kevin Hemenway	Caffeine: Nature's Sleeping Pill

Listing 3 contains a critical "database no-no": duplicate data. For every book I write, my name is repeated, meaning there are that many more chances for misspellings or similar errors to occur. And what happens if I suddenly become "Kevin Hemenway, Sr."? Suddenly, someone has to update four records before the correction is complete. This updating of multiple records can also bring the same possibility of corrupted data. Likewise, duplication of data is a waste of space: if I write a hundred books, you've got one hundred strings worth of "Kevin Hemenway", an after-school chalkboard experience that's just useless.

A relational database helps you store data just once: instead of a hundred Hemenway's running around on the moon, you'll only have one because you'll be relating a single record in the *person*'s table with a single record in the *book* table. Since the *relationships* are keyed toward the unique id of the tables, adding "Sr." to my name affects only that single row of that *person* table: the linkage defined in *relationships* remain unchanged. Listing 4 shows a partial (for brevity's sake) example of two of my books Listing 1's database.

Listing 4: An example of data in a relational database.

mactech.txt

Table: person

id	name	title	designation
1	Kevin Hemenway	Mr.	

Table: books

id	title
1	Mac OS X Hacks
2	Spidering Hacks

Table: relationships

person_id	book_id
1	1
1	2

Even though it's more tables and appears to be more "work", you've segregated and categorized your data so that it stands alone without duplication, and related it to one another via the unique ids assigned to each table. You can modify the data of a *book* without having to worry about corrupting the data in *person*, and vice versa.

RUNNING SQL COMMANDS AGAINST YOUR DATABASE

This is all fine and dandy, but we've still yet to send the SQL commands in Listing 1 to the *mactech* database we created last issue. To do so, make sure you've got your *mactech.sql* file nearby (the following steps assume it's located on your Desktop) and open a Terminal. Since we've got our commands in one file, we're going to send them to MySQL in one big batch. If you've not created a *.my.cnf*, be sure to add *-u username -p*:

```
~ > mysql mactech < ~/Desktop/mactech.sql
```

If there are no syntax errors in your SQL (like a missing comma, semi-colon, incorrect column type, etc.), then an error will be reported; otherwise, MySQL will succeed silently. So how do you *really* know if things went all right? Just like we confirmed our database creation with *mysqlshow*, we can pass the database name to get specific information, or even a table name to get more (output has been truncated, and again, beware of your username and password if you've not set them in *.my.cnf*):

```
~ > mysqlshow mactech
Database: mactech
+-----+
| Tables |
+-----+
| books  |
| person|
| relationships |
+-----+

~ > mysqlshow mactech books
Database: mactech Table: books Rows: 0
```


Field	Type	Null	Key
id	int(11)		PRI
title	tinytext	YES	
publication	date	YES	

Since we've yet to add any data into our database, the status of the **books** table reports 0 rows. Let's go ahead and add the data as seen in **Listing 4**. Open up a new text file called **mactech-insert.sql**, and transcribe the contents of **Listing 5** into it.

Listing 5: Adding data to our relational database.

mactech-insert.sql

```
INSERT INTO books
VALUES ("", "Spidering Hacks", "2003-11-01");

INSERT INTO books (publication, title)
VALUES ("2003-04-01", "Mac OS X Hacks");

INSERT INTO books
SET publication = "1998-03-01",
    title = "Caffeine Dreams";

INSERT INTO person
VALUES ("", "Kevin Hemenway", "", "", "Mr.", "");

INSERT INTO relationships VALUES (1,1);
INSERT INTO relationships VALUES (1,2);

UPDATE books SET publication = "2003-03-01" WHERE id = '2';
DELETE from books where title = "Caffeine Dreams";
```

There are a lot of different things happening here, so we'll step through each one. The first three statements are different ways of saying the same thing: "we want to insert this data into the **books** table." The first assumes you know, exactly and without fail, the order of the columns within your table. Three years from now, if you find yourself adding a new column between the **id** and **title**, this SQL statement will break. You'll also notice that the first column, being **id**, is left blank: this tells MySQL (as per our database definition in **Listing 1**) to insert the unique ID automatically, in this case "1" (since this is the first row we've added).

The second statement solves all the problems of the first by specifying, in any order, which columns the data should go in. The first column corresponds to the first argument to **VALUES**, the second column to the second, and so on. Since we're specifying columns by name, we don't need to mention **id** at all as MySQL will handle that for us. The third statement is yet another way of **INSERTing** data, and is my currently preferred method (see **UPDATE**, below).

The fourth statement is another **INSERT**, only into our **person's** table. I've left most of the fields blank, primarily so that it wouldn't word-wrap on the printed page. If they were defined, they'd just be dates and strings, something I'm sure all readers are familiar with.

The fifth and sixth columns insert IDs into our **relationships** table. Since I know that these were the first two books entered, I know what IDs they've received: 1, and 2, respectively. Once we get into interacting with MySQL with PHP and Perl (next article), I'll show you how to programmatically find out what IDs

just-added data receives (which is a much smarter way of doing things than hard-coding, as we do here).

The seventh statement shows how to update data you've already added. It is very similar to our third **INSERT** statement (the syntax I prefer) save for two changes: instead of **INSERT**, it's **UPDATE**, and we specify which rows to update with a **WHERE** clause. You'll be using **WHERE's** an awful lot with SQL as it's the primary method of selecting relevant results. Because the syntax of **UPDATE** is very similar to the syntax of our third **INSERT**, it's easy to minimize duplicated code within any of our programs. I use the logic in **Listing 6**... with this style, if the columns in **books** ever change, I only worry about one location (for both **INSERT** or **UPDATES**) and not two (necessary if we were using differing syntaxes).

Listing 6: Example logic to minimize code and SQL duplication

psudocode-for-update-inserts.txt

```
$sql = "";

if ($updating) { $sql = "UPDATE books SET "; }
elseif (! $updating) { $sql = "INSERT INTO books SET "; }

# assume variables are userinput
$sql .= "title = '$title', " .
        "publication = '$publication' ";

# same assumption with $id - user specified it.
if ($updating) { $sql .= " WHERE id = '$id';" }
```

The eighth and final SQL statement in **Listing 5** is an example of deleting a record, again determined by a **WHERE** clause. Be especially careful with **DELETE** statements: once your data is gone, it's "gone" gone.

To execute this batch of statements, we run the same command line as before, only with our new filename: **mysql mactech < mactech-insert.sql**. As before, the command will succeed silently, otherwise an error will be spit to the Terminal. You can verify that things went smoothly by checking the number of rows displayed in a **mysqlshow** statement: if there are two for **books**, two for **relationships**, and one for **person**, you did jim-dandy.

HOMEWORK MALIGNMENTS

Next issue, we'll talk about how to **SELECT** data from our database, as well as how to perform more **INSERT**, **UPDATES**, and **DELETES** within a programming language like PHP or Perl. If we have space, we'll throw together a few steps on installing applications like **phpMyAdmin** and **CocoaMySQL**. Until then, contact the teacher at morbus@disobey.com.

Self, don't forget to put funny stuff here.

Uhhh... should we leave this in? —Editor

Kevin said he'd email us. What about our deadline?

—Layout

He's never let us down before. He'll come through.

—Editor

By Chris Kilbourn

Life Change Ahead

Starting a business will change your life. How it will change is up to you.

You might be starting a business to become rich or to create a better world. Maybe you want to be in control of your schedule instead of being at the whim of some pointy-haired boss. Perhaps this is not your first go-round on the entrepreneurial ride, and you are hoping to finally achieve success this time around.

Stop for a moment and think about what your success will look like.

Are you pleasantly retired somewhere sunny with a drink in your hand while having your shoulders massaged? Are you donating the tons of cash you have earned to your favorite charities? Are you standing on the floor of the New York stock exchange ringing the opening bell on your initial public stock offering? Are you closing up shop after another great day at the office?

No matter your goal, it should be what is driving you to start a business, and it should be a clear goal. Beginning an enterprise is a huge undertaking that will commit and preempt your time and energy for months - if not years. If you are still a bit hazy about why you want to start a business, don't.

In this month's article we will examine in more detail the dreams and realities of starting and running a business, look at goal-setting, explore ways of integrating your business into your life, examine the larger hazards that you are likely to encounter as well as explore some of the rewards of business ownership.

Your life will change significantly after you start a business in ways intended and unintended. But if you know beforehand what you want from your business and what it will want from you, the inevitable hard knocks to come will land only glancing blows upon you instead of sending you down for the count.

MYTHS VS. REALITIES

Myth #1: Business owners are rich

Unless you are independently wealthy, you are working in order to earn money to pay for living expenses - and so are the vast

majority of business owners. For some reason, many people seem to believe that if you own your own business, you must be rich or at least well off enough to only be working at your business for 'fun.'

The American Dream, and the technology industry in particular, perpetuate this myth by lionizing its most successful entrepreneurs: Jobs, Gates, Dell, McNealy. They have earned their accolades. By only focusing on them, though, you lose sight of the thousands of people struggling in their businesses day-to-day, propped up by hope and personal credit, desperately trying to make that next sale to cover payroll.

Or to pay their mortgage.

Business ownership reality is that money is perpetually tight, and depending upon your business, it can attach to your personal finances like a leech. I have yet to meet an entrepreneur that has not been through a period of personal financial crisis due to one of their businesses.

The harsh reality is that most businesses fail because they run out of money and that before these businesses finally run out, the owner has pumped in many of their own dollars. In no small number of cases business owners are forced to declare personal bankruptcy when their businesses fold, having pledged personal credit guarantees to their creditors that they no longer can afford to pay.

In my own case, at one point in time I was juggling around \$500,000 worth of personal debt and credit guarantees. It made refinancing a car loan an extremely interesting exercise once. This is par for the course. Be prepared to have your personal credit record end up in a strange limbo land if you decide to self-finance your business.

The entrepreneur's dream is that there is a large payout somewhere along the line. Either as a steady cash cow like software upgrades or as a big hit like a killer piece of hardware or the sale of the business. For those of you starting your business for the big payday, realize that there are only two ways to get that money: as cash flow out of the business or by selling the company outright. Your business model and plan will guide you as to which will be appropriate. We will cover this in our next column.

Myth #2: Business owners have more free time

Business ownership: lounging by the pool with a martini and a cigar watching gardeners trim your hedges and clip your

Chris Kilbourn is an independent small business, network and web infrastructure consultant. Chris is also the founder of digital.forest, Inc., <http://www.forest.net>, which offers database, application and web hosting services in addition to server colocation. When he's not out running marathons, you may contact him at chrisk@forest.net.

grass while you try and decide between Switzerland and New Zealand for your next ski vacation. Not.

The reality is that business owners are likely to be working 16 to 18 hour days and likely have not had a vacation in a while. In my own case, I did not have more than three continuous days off in five years and I worked close 70 hours per week for the first two years with some 100 hour weeks thrown in for good measure.

Businesses require huge investments of time, and the initial startup phase will require the most. There are business plans to write, people to call, office space to secure, telephone lines to order, staff to hire, equipment to secure, etcetera, etcetera, etcetera. Then you still have to make time to do the **work** of your business. What you will have is more flexibility over your own schedule. Unless you have business partners, there will be no one to tell you that you are late to work or that you cannot take a morning off to play hooky now and then.

If your goal is to start a business to enable to you to have a flexible schedule with more free time for yourself, be aware that it may take several years before your business has grown enough to allow you to delegate projects and responsibilities to your staff, freeing you up to do other things in your life.

Myth #3: Business owners have no boss

It is true that when you own your own business, there is no one person to tell you what to do. What gets short or no shrift in the daydream of working for yourself is that each and every customer of yours will be your boss.

Think about it for a moment. If you are currently working for someone else, that person provides you with your paycheck and directs you in the tasks that you do. Customers also pay you, albeit indirectly, and they can drive what it is that you do on a daily basis.

This mode of work can take some getting used to. Instead of one person that you develop a working relationship with over time, imagine tens, hundreds, even thousands of people that all want you to do something for them and claim a bond on your time. While not a codified law, I have come to believe that there is an inverse proportion to the level of customer demand versus the amount of money being exchanged!

With a myriad of things to do and only a limited amount of time to get them done in, time and project management skills are critical to the success of your business. Working for someone else generally allows you the luxury of not having to worry too much about what work you should be doing and when you should be doing it. If your personal skills are rusty or lacking in these two areas, I would highly recommend taking a class or investing in a device or software that will help you keep track of the things you need to do, and when you need to do them.

Myth #4: Business owners are naturally successful people

For the reality check on this, attend a local venture capital or entrepreneurial event and ask the other attendees about their past failures. Chances are, you will receive an impassioned story about the one (or many) ventures that did not work out for them. Most businesses fail. It is just the way natural selection in the economic environment works.



ENGINEERING BUSINESS SOLUTIONS

- SINCE 1989 -

**PREMIER REPORT SOLUTIONS
FOR MAC OS X**

VVI®

www.vvi.com

info@vvi.com

888-VVI-PLOT

Many entrepreneurs fail several times before 'striking it rich,' even passing through bankruptcy several times. Stories like these are what fuel the 'rags to riches' archetype.

Success in business is not guaranteed, even with maximum effort. Perseverance is key. Expect that there will be large setbacks. Expect that there will be some failures along the path. This does not mean you should accept a defeatist attitude - on the contrary - keep focused on the goal and keep trying, for perseverance will eventually guide you to where you seek to be.

Myth #5: Business owners get to do exciting things

If cleaning out the company refrigerator because the onion dip went bad or staying up for 72 hours straight troubleshooting balky technical equipment is your idea of excitement, you are going to love running a business!

The myth is that you will be out there cutting deals and changing the world one customer at a time, all the time. The reality is that you will spend most your time in meetings, doing presentations, managing staff, sorting out troubles of all stripes and dealing with the crisis of the hour. Oh, and cleaning the fridge, unless you can afford a cleaning service.

There is excitement in running your own business, but just like everything else in life, the time in between the exciting bits tends to be filled with housekeeping duties and routine tasks that help to create a space for the excitement.

ARE YOU DREAMING?

One of the most satisfying things about starting a business is working towards making a dream come true. The funny thing about dreams are that they can crowd out objective reality, blind you to peril, and isolate you in your own happy little world to the exclusion of everyone and everything else. Entrepreneurs immerse themselves into their dream in order to actualize it. If you follow this path, it is important to come back to the real world now and then.

So how do can you tell when your dream is beginning to tip towards a nightmare? If things are going poorly, chances are that friends and family are already sounding a klaxon in your face, attempting to gain your attention. Ignore it at your personal peril.

It will be up to you to do a gut-check to see if the fears and concerns they have about you or the business are real or imagined. In a perfect world, friends, family and lovers will support you with unconditional love and understanding while you pursue your dream.

In the real world, friends get annoyed because you really don't have time to listen to them whine about their golf game when you have invoices to get out, family gets miffed that you missed a birthday because you were working an 80 hour week to get a major proposal out the door and your lover becomes enraged when they realize you have more passion for your business than you do for them.

Somewhere within and amongst the four worlds you all concurrently share, (your dream world, your reality, their dream world and their reality,) there is some semblance of what is really going on, that is both good and bad.

Everyone's circumstances are different but as sweeping general guidelines attempting a third mortgage on your house to

pay business bills is bad; trying to figure out how to invest profit is good; missing sunset walks with your significant other in order to compile software is bad; having time to take a three hour lunch with your significant other is good.

Separating large issues from small ones can be tricky though. Is your spouse's concern about money due to a real financial shortfall or have you just gotten around to depositing a check? Are you working too much and becoming less effective as you go or are you getting all sorts of things done that others cannot see?

In any event, communicate with those around you. That includes talking and listening. Remember that what you are doing may be terrifying to even your closest loved ones and that their fear for you can amplify and distort small and large issues.

I will leave it as an exercise to the reader to triangulate their own situation.

SETTING CLEAR AND ATTAINABLE GOALS

OK, now you know that things will not be easy and that you will need to remain vigilant when it comes to avoiding becoming entangled in a fantasy of how things should be with your business. One of the best ways to stay grounded is to have clear and attainable goals.

Before I founded digital.forest, I thought quite a bit about what I was hoping to accomplish. It was time well spent, as it provided me with something to strive for when things became difficult and I was tempted by despair. I set incremental goals that led to a larger goal which I hoped to achieve. By attaining each incremental goal in turn, I secured additional confidence to push forward and look back upon when it seemed like the next step was out of reach.

As an example, one of my personal goals was to have a more flexible schedule. I wanted to be able to take time off when I wanted or needed to and have other people take care of the business when I was not there.

When I first started out, I was a one-man show, so I knew that it was unrealistic to expect that I could just take time off willy-nilly, as there was all sorts of work to be done. In fact, there was so much work, I was regularly working more than 80 hours a week for the first several months of the business.

The incremental goals I set for myself were to ratchet down my hours over the weeks, and to take a set amount of personal time each week to do something non-business. As the business began to grow and I was able to hire my first employee, I was able to off-load some pager duty to her, which allowed me some control over my sleep every other week for the first time in a couple of years.

As the business continued to grow and there were more employees to delegate tasks to, I was able to reach a point where I could take a real vacation and not have to worry about being called back to the office or be interrupted by a telephone call. There were the invariable setbacks when it seemed like I was back to living at the office and working non-stop, but those periods became shorter and shorter as time went on.

It was another year before I could juggle my schedule enough to take the odd day off to go skiing every now and then in the winter. From working insane hours to being able to take a normal vacation and the odd day off took me six years.

Each step along the way, I knew what it was I was working towards, and I always made the next step a little harder but not impossible to get to from where I was at the time.

Some of the other goals that I set for myself were to expand my personal career skill set by learning how to do new things, have the opportunity to meet a diverse group of people and create a profitable business.

Granted, these goals were a long way from say, curing cancer, but none of them were unrealistic, and I have the personal satisfaction of having achieved them.

Spend time thinking about what personal goals you would like to achieve with your business. The more effort you put into this and the clearer they are, the easier it will be for you to articulate your vision to others, and they can be your personal lodestone when feeling a bit lost about the direction of your next step.

WORK TO LIVE OR LIVE TO WORK

Adam Smith's invisible economic hand may guide us to start a business, but the hand has nothing to say about why we work so hard to do so. Reading Max Weber or Ayn Rand might provide us with an insightful glimpse here and there, but in the end they only provide us with archetypes to study and ponder.

Western capitalistic society is deeply committed to certain notions about work. Some of these beliefs are tacit and unspoken, others are common idioms that we take for granted.

Wheel and deal. Nose to the grindstone. Money talks.

In many cases, how you view, approach and perform your work is a reflection of your culture, your upbringing and your values. Some cultures value family time over business time, while another might view work as family time.

No matter your background or beliefs, be sure to ask yourself these three questions as you plunge into your business:

- Is what you are doing meshing with and advancing you towards your goals?
- Are you building in enough time in your schedule for yourself, your family and your friends?
- Are you having fun?

These three questions speak to self actualization, of staying connected to others and to joy. Your answers to these questions will ebb and flow over time. Some times they will be easy to answer and at other times, devilishly difficult.

If you are running a business and are having trouble getting to 'yes' on any of these questions, it is time for you to stop, step back, and re-evaluate what you are doing with your time.

Integrating a business into your life can sometimes be harder work than getting the business off of the ground in the first place. If the business is unduly encroaching into your life, you will need to systematically remove it from the areas into which it is intruding.

Setting clear boundaries between work and personal life should be the first place to start. At one point in time, I was carrying a pager 24 hours a day, 7 days a week and before I went to bed and when I arose in the morning I would check my email for problems that needed to be troubleshoot. It was not a

healthy time for me, as I felt like I never had a spare moment away from the business. I was at 'no' for question 2 above.

I worked like that because I was still at 'yes' on the other two questions. Hiring staff relieved that time pressure on me and allowed me to enjoy my work even more.

DANGER AHEAD

When I hit the trifecta of feeling like I was not making progress towards my goals, when I did not any have time for myself, my family or my friends and when I was not having any fun in the business, I was a wreck.

The Fall of 2000 and the Winter of 2001 was a rough period of time for many in the Internet industry, and digital forest was not immune to the shrinking Internet bubble. It created challenges and stresses like I had never before experienced. Those pressures, coupled with a deteriorating relationship in my personal life, left me unable to sleep, gave me chest pains, added 30 pounds to my frame, inserted searing migraines into my head and turned more than a few hairs on my head prematurely gray.

I was a classic example of a burned out basket case.

By March of 2001, my life consisted of a haze of non-sleep at night and non-functioning at work. I ignored my friends, my family, my hobbies, what was going on in the world around me. I was perpetually stressed about a lack of money for the business and the crushing weight of debt that would land on my head if the business failed.

So I quit. I stepped down as CEO of the company I had founded and took a four month sabbatical. A great weight was lifted from me by doing so, and I felt better emotionally than I had in years.

I relate this tale as it neatly illustrates what can happen if you lose focus on your goals. Like many others during the same period of history, I allowed myself to be swept into the group hallucination that markets would ever expand and that simply having a pulse in the Internet industry was a guarantee of riches.

Not more than six months prior I had felt on top of the world. I had raised close to a million dollars for the company, business was booming and it seemed like the sky was the limit.

But by taking on the dreams and goals of others, I had forsaken my own and it left me vulnerable to the vicissitudes of the marketplace and the challenges of managing a business during a downturn in the economy.

Proof that the greatest threats to our businesses lay not with our competition, but within ourselves.

PARADISE ATTAINED

So what are your criteria for happiness? For satisfaction? For riches?

If you know them in your daily life, you will also find them in your business. If you seek the answers to these questions within your business, be prepared for a rocky ride.

Remember: how your life will change is up to you, but you must first decide what you want to change.

Next month: We will lighten up on the metaphysics and look at the concrete world of planning. It is good for SCUBA diving, hiking, and business.

Andrew Troelsen, Minneapolis, Minnesota

The SSCLI: Managed Languages, Samples and (more) Programming Utilities

Exploring .NET development on Mac OS X

PATCHING THE SSCLI ON PANTHER

Before we dig into the meat of this month's installment, allow me to address an issue regarding the process of building the SSCLI. Like myself, you may have purchased and installed Mac OS 10.3 (Panther) on your development machines. As you are aware, Panther upgrades numerous core Unix tools, including gcc (the GNU Compiler Collection). Given these changes, the Shared Source CLI will *fail* to build on new Panther machines, while upgrading an existing installation of Mac OS 10.2 to 10.3 may cause certain SSCLI command tools to become 'funky' (e.g., fail to execute).

Lucky for us, various independent patches do exist (as of today, no official patch from Microsoft exists, but check the SSCLI web site every now and again). The patch that I used to rebuild the SSCLI after installing Panther was developed by Dr Nigel Perry at the University of Canterbury. Now, given that your machine may have specific customizations, I cannot guarantee that the path I took to patch the SSCLI will be identical to your own. However here are the steps that worked for me:

- Download the patch (rotor_mark2_diffs.txt.gz) from <http://www.mondrian-script.org/rotor/>
- Run the gunzip utility to extract the patch file (panther_mark2_diffs.txt) and copy it to /sscli directory
- Apply the patch by changing to your /sscli directory and run the following Terminal command (**Listing 1**)

Listing 1. Patching the SSCLI on Panther

```
patch -pl <panther_mark2_diffs.txt
```

At this point, you can rebuild the SSCLI as described in the November issue. It is worth noting that I needed to issue the './buildall' command more than once to fully rebuild the entire

code base (while selectively ignoring non-critical build-errors). In any case, hopefully these instructions will do the trick. With this out of the way, let's dig deeper into the SSCLI.

SELECT SAMPLES OF THE SSCLI

As mentioned in a previous article, the SSCLI ships with various sample applications that illustrate numerous aspects of the CLI (Common Language Infrastructure) and Base Class Libraries (BCL), the full list of which can be viewed by opening /sscli/samples/samples_index.html. In this installment, I'll introduce you to some (but by no means all) intriguing sample applications, specifically:

- The JScript.NET, cLisp and myC compilers
- Codetohml, corcls and typefinder
- Pigpad

Understand that the goal of this issue is *not* to dive into all the gory details of the underlying technologies surrounding these samples. Rather, here we will learn about a number of samples that either (a) explore the language and platform agnostic aspects of .NET or (b) showcase useful developer tools. Rest assured that the technologies used in these sample applications (such as reflection and dynamic code generation) will be explored at a later time.

Before we begin, recall that the /sscli/samples directory contains a majority of the SSCLI sample code. You may directly view, modify and compile these files at the command line using csc (detailed in the December issue). Also recall the SSCLI sample applications are compiled when you build the SSCLI and are placed under the /sscli/build/v1.ppcfstchk.rotor/samples directory. These applications are ready to run at the Terminal (or xterm) using the clix application launcher. Pick your poison, enable an SSCLI Terminal session, and read on.

Andrew Troelsen is a seasoned .NET developer who has authored numerous books on the topic, including the award winning *C# and the .NET Platform*. He is employed as a full-time .NET trainer and consultant for Intertech Training (www.intertechtraining.com) who apologizes for failing to submit his January article in time. You can contact (or scold) Andrew at atroelsen@mac.com.

DISTRIBUTE SECURE SOFTWARE



- > CHOOSE YOUR PATHS.
- > NAIL YOUR TARGETS.
- > REAP THE REWARDS.

Privilege™
SECURE SOFTWARE COMMERCE

SECURE SOFTWARE COMMERCE for ESD, CD-ROM, Casual Sharing and Media-based distribution
Increased revenues. Reduced costs. Measurable results. That's the power of Privilege. Feel it for yourself.
For your FREE Evaluation Kit, call 1-800-562-2543 or visit GoPrivilege.com/YourWay.

N. America: 1-800-562-2543, 847-818-3800 or Priv.us@eAladdin.com **Int'l:** +972-3-636-2222 or Priv.il@eAladdin.com
Germany: Priv.de@eAladdin.com **UK:** Priv.uk@eAladdin.com **France:** Priv.fr@eAladdin.com
Benelux: Priv.nl@eAladdin.com **Japan:** info@Aladdin.co.jp

©2004 Aladdin Knowledge Systems, Ltd. Aladdin, Aladdin Knowledge Systems, Ltd., and HASP are registered trademarks of Aladdin Knowledge Systems, Ltd.

Aladdin™
SECURING THE GLOBAL VILLAGE
eAladdin.com

EXPLORING THE ALTERNATIVE 'MANAGED LANGUAGES' OF THE SSCLI

In the .NET universe, the term *managed language* refers to a programming language which is .NET aware; meaning, the corresponding compiler (a.k.a. a *managed compiler*) has been updated to understand the .NET platform. Out of the box, the SSCLI ships with a total of five managed compilers (two of which are research-oriented in nature):

- The C# compiler (csc)
- The raw CIL compiler (ilasm)
- The JScript.NET compiler (jsc)
- An experimental LISP-based .NET compiler (cLisp)
- An experimental C-based .NET compiler (myC)

Given that the previous issue examined the basic details of working with csc and ilasm, let's take some time to examine the three remaining managed compilers of the SSCLI.

The JScript.NET Compiler

The JScript.NET compiler allows you to build .NET solutions using the syntax of the JScript (a.k.a. ECMA-script) language. Unlike traditional JScript however, the JScript.NET compiler (jsc) transforms the input *.js files into a true-blue .NET assembly rather than a simple blurb of code to be processed by a scripting engine. In addition to this compelling change, JScript.NET also supports the following enhancements to the original ECMA 262 specification:

- Support for strongly typed data
- Full-blown object-orientation
- Support for type partitioning via 'packages'
- Complete access to the .NET base class libraries

The JScript.NET compiler that ships with the SSCLI is very interesting in that the jsc code base (located under /sscli/jscript) is authored entirely in C#! Thus, given that jsc is in fact a .NET assembly, it must be executed via clix.

To view each of the options provided by the JScript.NET compiler, change to the directory containing jsc (/sscli/build/v1.ppcfsthk.rotor) and issue the following command from an SSCLI-enabled Terminal (**Listing 2**):

Listing 2. Launching the JScript.NET Compiler via clix

```
clix jsc
```

As you can see from the resulting output, many of the options provided by jsc are identical to those of csc. For example, jsc supports /output, /target and /reference flags as well as support for response files. One feature of jsc is unique however, and is described in **Table 1**.

The interesting option of jsc	Meaning in Life
autoref	This option (which is enabled by default) informs the compiler to automatically reference external assemblies based on the list of import statements in your *.js code files.
	Using this feature, you are not required to explicitly specify assemblies using the /r flag (as you would need to do in C#).

Table 1. The interesting flag of the JScript.NET compiler

While this is not the place to examine each and every aspect of the JScript.NET programming language, **Listing 3** provides some sample code to use throughout this article:

Listing 3. SimpleJScript.js

```
// 'import' is a JScript.NET specific
// keyword use to reference .NET namespaces.
import System;

// A simple class type.
class Person
{
    // State data of the type.
    private var mName : String;
    private var mAge : int;

    // Type constructor.
    function Person(name : String, age : int)
    {
        this.mName = name;
        this.mAge = age;
    }

    // A read-only 'property'
    function get Name() : String
    {
        return this.mName;
    }

    // Read-write property.
    function get Age() : int
    {
        return this.mAge;
    }

    function set Age(newAge : int)
    {
        if ((newAge >= 0) && (newAge <= 130))
            this.mAge = newAge;
        else
        {
            Console.WriteLine("Ug...bad value...");
        }
    }
}

// Prompt user for information about this person.
Console.WriteLine("***The Amazing JScript.NET App***");
Console.Write("Enter person's name: ");
var name : String = Console.ReadLine();
Console.Write("Enter initial age of person: ");
var age : int = int.Parse(Console.ReadLine());

// Now create a person using input.
var myPerson : Person = new Person(name, age);

// print name using intrinsic JScript f(x).
print(myPerson.Name);
```



```
// Base class library call.
Console.WriteLine(myPerson.Age);

// Change age with Age property and reprint.
chucky.Age = 26;
Console.WriteLine("{0} is now {1} years old.",
    myPerson.Name, myPerson.Age);
Console.WriteLine("All done. See ya");
```

The first point of interest is the JScript.NET 'import' keyword, which is similar in function to the C# 'using' keyword in that it allows you to specify the name of the .NET namespaces this file is manipulating.

Next is the 'class' keyword of JScript.NET used to define a .NET class type named 'Person'. Note that type member variables are defined using the 'var' keyword and the following syntax (**Listing 4**).

Listing 4. Defining Type Member Variables ala JScript.NET

```
class Person
{
    // AccessModifier var NameOfVariable : DataTypeOfVariable;
    private var mName : String;
    private var mAge : int;
}
```

Given that these data points have been defined as private, we need to provide a manner in which the object user can safely manipulate these values. Under the .NET platform, encapsulation services are provided (in part) by the use of type *properties*. Simply put, properties are get (accessor) and set (mutator) methods in disguise. Here, our Person class defines a read-only property termed Name and a read/write property named Age.

In JScript.NET, properties are declared using the 'function get' and 'function set' keywords. Like a traditional accessor/mutator pair, properties are used to control access to state data. In the case of our read-only Name property, we are returning a copy of the private mName member variable (**Listing 5**).

Listing 5. A Read-Only Property ala JScript.NET

```
// Note that properties must be defined
// with regard to the type of data they
// encapsulate.
function get Name() : String
{
    return this.mName;
}
```

The Age property is a bit more interesting in that the function set method performs simple data validation on the incoming parameter (**Listing 6**).

Listing 6. A Read/Write Property ala JScript.NET

```
function set Age(newAge : int)
{
    if ((newAge >= 0) && (newAge <= 130))
        this.mAge = newAge;
    else
    {
        Console.WriteLine("Ug...bad value...");
    }
}
```

The major benefit of using .NET properties, over accessor/mutator methods such as GetAge() and SetAge(), is the fact that it simplifies the syntax for the object user. Thus, rather than authoring the following Java-like code (**Listing 7**):

Listing 7. Traditional Get/Set methods

```
// Encapsulation using accessor/mutator methods.
myPerson.SetAge(30);
var theAge : int = myPerson.GetAge();
```

We can simply write the following (**Listing 8**):

Listing 8. Properties in action

```
// Encapsulation using .NET properties.
myPerson.Age = 30; //Triggers set functionality.
var theAge : int = myPerson.Age; //Triggers get functionality.
```

Notice how properties provide the illusion that you are accessing a point of public data (which would be bad insofar that that would break encapsulation) while still honoring your data validation logic.

The final point is the block of 'freestanding' code at the end of the SimpleJScript.js file. Notice that we are blending the use of the .NET System.Console type as well as the JScript 'print' function to print out the values of the person to the Terminal window (**Listing 9**).

high quality - competitive rates - 16 years experience - award winning

Full Spectrum Software Development & Testing

Device Drivers
Carbon Cocoa
Real Basic

Rescue Missions

Cross Platform Development

1661 Worcester Road
Framingham, MA 01701

508-620-6400

www.FullSpectrumSoftware.com

competitive rates - 16 years experience - award winning - high quality

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

Listing 9. Blending BCL types with intrinsic JScript function calls

```
// Now create a person using input.
var myPerson : Person = new Person(name, age);

// print name using intrinsic JScript f(x).
print(myPerson.Name);

// Base class library call.
Console.WriteLine(myPerson.Age);
```

Compiling and Running our JScript.NET Application

Assuming you have saved the previous code to a file named SimpleJScript.js, you can compile this file into a .NET code library with the command shown in **Listing 10** (assume the input file is in the same location as jsc. If not, supply the full path):

```
clix jsc SimpleJScript.js
```

Listing 10. Compiling our *.js file

You can now run your assembly as you would expect (**Listing 11**).

```
clix SimpleJScript
```

Listing 11. Launching the JScript.NET compiler via clix

Figure 1 illustrates one possible session.



Figure 1. SimpleJScript.exe at work.

Dissassembling Our JScript.NET-Based Assembly

If you have been playing around with the C# programming language, it may surprise you that JScript.NET allows programmers to define programming logic not contained within a type definition. C# (as well as most other .NET-aware languages) demand that all programming logic be contained within a type or type member definition. In reality, the same constraint is found with JScript.NET, however the compiler will take care of the details on your behalf. To illustrate, load your SimpleJScript.exe assembly into ildasm (**Listing 12**).

```
ildasm SimpleJScript
```

Listing 12. Dissecting SimpleJScript via ildasm

As you look over the output, you will find that all of the global code has been automatically placed in a method named (surprise, surprise) 'Global Code', which is defined within a type named 'JScript 0'. This class type extends

Microsoft.JScript.GlobalScope, which is defined within a helper assembly named Microsoft.JScript.dll (located under the /sscli/build/v1.ppcfstchk.rotor directory). **Listing 13** shows the CIL of the type in question (edited for clarity).

Listing 13. The auto-generated 'JScript 0' Class Type

```
.class public auto ansi 'JScript 0'
    extends [Microsoft.JScript]
    Microsoft.JScript.GlobalScope
{
    .method public instance object
        'Global Code'() cil managed
    {
        // All of your global code is placed here...
    }
}
```

In addition to the 'JScript 0' type, jsc also generates a class named 'JScript Main' that defines the application's entry point: Main(). The Main() method leverages several types within the Microsoft.JScript.dll assembly to execute the program. **Listing 14** hits the highlights.

Listing 14. The auto-generated Main() method

```
.class public auto ansi 'JScript Main'
    extends [mscorlib]System.Object
{
    .method public static void Main(string[] A_0)
        cil managed
    {
        .entrypoint

        // Define a variable of type GlobalScope.
        .locals init (class [Microsoft.JScript]
            Microsoft.JScript.GlobalScope V_0)

        // Create the 'JScript 0' object.
        IL_001e: newobj instance void
            'JScript 0'::.ctor(class [Microsoft.JScript]
                Microsoft.JScript.GlobalScope)

        // Now call the 'Global Code()' method.
        IL_0023: call instance object
            'JScript 0'::'Global Code'()
        IL_0028: pop
        IL_0029: ret
    }
}
```

This wraps up our exploration of the JScript sample application. If you are interested in exploring JScript.NET in greater detail, hundreds of articles and sample applications can be found online (just be sure to have a space between JScript and .NET when searching [JScript .NET not JScript.NET]).

The cLisp Compiler

The next sample application (cLisp) is a partial LISP compiler that targets the .NET platform. As mentioned, cLisp is more research-orientated in nature, in that it does not implement every aspect of the LISP programming language. However, the code base does provide an excellent foundation for those of you who are interested in extending cLisp with new functionality. As well, if you wish to build your own managed compiler that can function within the SSCLI, cLisp is a solid example to build upon.

Like the JScript.NET compiler, the cLisp compiler is authored in C#. If you navigate to the cLisp directory (/sscli/samples/compiler/clisp), you will find all of the necessary source code. As well, the cLisp directory contains a few *.lisp code files which can be used as input to the cLisp application. For example the fibo.lisp file computes a Fibonacci sequence (**Listing 15**).

Listing 15. fibo.lisp

```
(defun Fib (N)
  (if (<= N 0)
      0
      (if (= N 1)
          1
          (+ (Fib (- N 1)) (Fib (- N 2)))
      )
  )
)

(defun Fibo (N) (do ((count 0 (+ count 1))
                  (Fibo (car (Fibo 0)) (cons Fibo (car (Fibo count)))))
  ((> count N)
   Fibo)))

(Fibo 25)
```

Now, if you can't make heads or tails of the previous LISP code, don't sweat it. The point here is not to examine the syntactic details of LISP, but rather understand how LISP source code can be used to build a valid .NET image. To illustrate, navigate to the /sscli/build/v1.ppcfsthk.rotor/samples directory and compile fibo.lisp into an executable assembly using the following command (**Listing 16**).

Listing 16. Compiling *.lisp files via cLisp

```
clix clisp fibo.lisp
```

The resulting .NET assembly (fibo.exe) may now be launched using clix. **Figure 2** shows the program's output.



Figure 2. Fibo.exe at Work.

Much like jsc, cLisp will auto-generate a class type and program entry point behind the scenes (this can be verified by loading Fibo.exe into ildasm). While this example is enticing, understand that given the fact that cLisp is an *illustrative* compiler, you do not have access to the BCL types.

A Brief Word Regarding the myC Compiler

Last but not least, it is worth pointing out that the SSCLI also ships with another research-oriented managed compiler named myC. This sample exposes a (very) limited subset of the C programming language to the .NET universe. Like cLisp, the sample code further illustrates the process of building a managed

New

PrimeBase 4.2

Replication Server

Check out the fully programmable Replication Server

- Bidirectional Updates supported
- Update 3rd-party DBMS
- Send Emails
- Post/get Data to/from Websites

SQL-Runtime Plugin
for REALbasic
\$ 499,-

All PrimeBase Server Software

- SQL-Runtime Libraries available
- SQL Database Server
- Application Server
- Replication Server
- Open Server

Available on the most popular platforms

- Completely cross-platform
- Full-text searching and indexing
- Mac OS & OS X
- Linux
- Solaris
- IBM AIX
- all Windows platforms

 **PRIMEBASE**

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com
e-mail: info@primebase.com
Fon: ++49 (40) 389 044-0
Fax: ++49 (40) 389 044-44

compiler. Sscli/samples/compiler/myc/myc.html describes the C# source code files used to build this managed compiler and I'll leave it to you to dig into the details if you so choose.

A GAMUT OF .NET PROGRAMMING LANGUAGES

The .NET platform may be manipulated using numerous programming languages, far beyond those that ship with the SSCLI. In the initial issue of this series (October, 2003) I listed a subset of .NET-aware compilers. If you wish to see a more all-encompassing list, navigate to <http://www12.brinkster.com/brianr/languages.aspx>. Here you will find links to compilers as diverse as Ada, Forth, Fortran, Java, LOGO (yes, I am not kidding), Mixal and Ruby as well as fully functionally C and LISP compilers.

Keep in mind however that many of these offerings target production level .NET distributions such as Mono, Portable .NET and Microsoft's CLR, and therefore may require a bit of tweaking to execute under the SSCLI. In just a few short issues, we will move from the research-oriented world of the SSCLI to a production-level distribution of the CLI named Portable .NET. At that time I will revisit the use of 'alternative' .NET programming languages.

CODE DOCUMENTATION AND TYPE ANALYSIS SAMPLES

Now that you have had a chance to check out the compiler-centric examples of the SSCLI, we'll examine samples of the code-formatting / assembly-investigation ilk. Each of the examples we will examine here are found under the /samples/utilities subdirectory. First up, codetohtml.exe

The CodeToHtml Sample Application

The codetohtml sample application (which is written in *JScript.NET*) generates HTML files representing C# or JScript.NET source code. To illustrate codetohtml in action, supply the SimpleJScript.js file created earlier as a command line argument (**Listing 17**).

Listing 17. Processing SimpleJScript.js
 clix codetohtml SimpleJScript.js

Figure 3 shows the default HTML format.



Figure 3. An HTML view of SimpleJScript.js

The runtime behavior of codetohtml is controlled by a set of initialization files, whose exact names are based on the language of the input file (for example, js_codetohtml.ini for JScript.NET and cs_codetohtml.ini for C#). Table 2 describes the role of each *.ini file.

CodeToHtml Initialization File	Meaning in Life
<lang>_codetohtml.ini	This is the core file which specifies the names of the other three related *.ini files as well as the associated style sheet to use when generating the HTML file.
<lang>_keywords.ini	Contains a list of all active keywords for the given language.
<lang>_futuresreserved.ini	Contains a list of reserved keywords for the given language.
<lang>_userdefined.ini	Contains a list of user-defined types (such as class names) to be recognized during the formatting process.

Table 2. The initialization files of codetohtml

Of course, each of these files may be modified to control the format of the resulting HTML. To begin, open up js_codetohtml.ini using your text editor of choice (**Listing 18**):

Listing 18. Contents of js_codetohtml.ini
 defstylesheet=codeblue.css
 keywords=js_keywords.ini
 userdefined=js_userdefined.ini
 futuresreserved=js_futuresreserved.ini
 replace=js_replace.ini
 language=JScript

Note that the default style sheet is named codeblue.css. This may be changed to the other sample style sheet, codewhite.ccs, by updating the defstylesheet value. If you examine either file, you will find a set of styles used to format each token category (keywords, code comments, user defined types and so on). **Listing 19** shows a partial listing.

Listing 19. Partial listing of codeblue.css
 div.code
 {
 background-color: rgb(0,0,128);
 font-family: "Lucida Console", "courier new", courier;
 color: rgb(255,255,0);
 font-size: x-small;
 padding: 1em;
 margin: 1em;
 }
 ...
 div.code span.userdefined
 {
 color: rgb(51, 51, 0);
 font-weight: bold
 }
 ...

To spice things up a bit, edit js_codetohtml.ini file to point to the codewhite.css style sheet. Next, modify js_userdefined.ini to recognize the System.Console type of

the base class libraries by adding Console to the end of the list. At this point, run codetohtml once again specifying SimpleJScript.js as the input file. Notice in **Figure 4** that each occurrence of the Console type is rendered using the 'userdefined' style.



Figure 4. Custom code to HTML formatting

Beyond creating custom style sheets for use by codetohtml, you will also want to check out the JScript.NET source code for this example (codetohtml.js) as it illustrates a number of interesting .NET base class library types. As you already know how to interact with the JScript.NET compiler, feel free to hack away to your heart's content.

The corcls and typefinder Sample Applications

Corcls and typefinder are essentially object browsing utilities (written in C#) that provide a lightweight alternative to ildasm. The corcls utility can be helpful when you simply wish to view the C# definition of a given type, rather than the underlying CIL code. If you run corcls at the command line (via

clix), you will see a list of all available options. **Table 3** documents a partial listing.

Interesting Options of Corcls	Meaning in Life
-w	Generates output in HTML format.
-m <i>nameOfAssemblyToSearch</i>	By default, corcls only searches a core .NET BCL assembly named mscorlib.dll. If you wish to browse other assemblies (including your custom assemblies), use the -m flag.
-noinherit	Prevents the display of inherited methods and fields.
-v	Displays only output visible (public and protected) classes, methods, or fields.
-p	Displays only output public classes, methods, or fields.

Table 3. Various flags of corcls

Again, by default, corcls only searches the core .NET assembly, mscorlib.dll. If you wish to view definitions for types defined in other assemblies, you will need to specify the '-m' option. By way of a few examples, **Listing 20** will display type information for System.String, **Listing 21** displays the same type as HTML (via the '-w' option) and **Listing 22** dumps information about the Person type defined within SimpleJScript.exe (originally defined in JScript.NET). Be aware that when you run load external modules with the '-m' option, you will need to ensure a copy of the binary file is in the same directory as corcls itself.

clix corcls System.String

Listing 20. Viewing System.String

Valentina

Object-Relational SQL Database

The fastest database engine for MacOS/Windows

It operates 100's and sometimes a 1000 times faster than other systems

www.paradigmasoft.com

Hosted by macserve.net

Download full featured evaluation version

EXTREME^{to} the MACS!

Aria Extreme

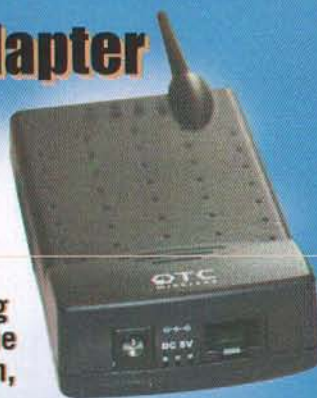
\$78.95



Add AirPort Extreme to any G3 or G4 PowerBook! Get blazing speeds up to 54 Mbps (five times AirPort Speed!), comes with an external stub antenna for better range than internal cards! (requires Mac OS X 10.2.6 or above)

**OTC Wireless
802.11g Ethernet Adapter**

\$164.95



Driverless Ethernet to 802.11g converter. Simply plug into the Ethernet port of any Macintosh, PC, or even printer to put it transparently on your 802.11g (54 Mbps) network! Use with a hub to enable multiple devices with one adapter!

DEV DEPOT[®]

www.devdepot.com

877-DEPOT-NOW

**Use AirPort
Extreme with
any Macintosh!**

Toll Free: 877-337-6866 • Outside US/Canada: 805-494-9797 • Fax: 805-494-9798 • orders@devdepot.com
PO Box 5200 • Westlake Village, CA 91359-5200

AirPort Extreme (802.11g) PCI Card

\$98.95



Easy to install! Step up to
AirPort Extreme (54 Mbps)
for any PCI Macintosh!
(requires Mac OS X 10.2.6 or above)

AirPort Extreme

AirPort Extreme Omni Antenna 5 dBi

\$78.95



Want more range from your AirPort
Extreme base station? This 5 decibel
antenna is 30% more powerful than
other brands!

We also offer a full line of AirPort (802.11b) products!



**Mac Wireless
USB
to 802.11b
Converter**



**AirPort PCI
to 802.11b
Converter**



**AirPort 802.11b
PCMCIA Card**



**Belkin Wireless
Cable/DSL Gateway
Router**

\$78.95

\$128.95

\$88.95

\$112.95


```
clix corcls -w System.String
```

Listing 21. System.String as HTML

```
clix corcls -m SimpleJScript.exe Person
```

Listing 22. Viewing Person

Figure 5 shows the result of executing Listing 22.

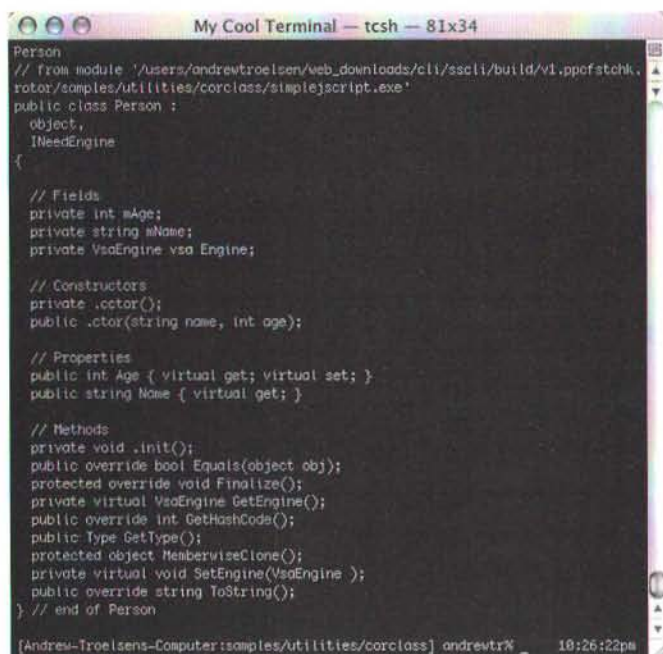


Figure 5. Viewing Person in the syntax of C#

The functionality of typefinder is similar to that of corcls, however this tool allows you to get much more specific. Using a set of display options, you can view various aspects of a type such as the set of interfaces it supports, the methods it contains and so forth. I'll allow you to explore the full set of options via the `findtype.html` file located under `/sscli/samples/utilities/typefinder`, however to prime the pump, Table 4 lists some possibilities.

Typefinder Option	Meaning in Life
-d: NameOfDirectory	Used to specify a directory which contains the assembly you wish to investigate.
-x	Used to specify the fully qualified name of the type in a given assembly.
-a	Shows all information for a given type, including inherited members (very helpful).
-e, -f, -i, -m, -p	Shows the events, fields, interfaces, methods or properties of a given type. These flags may be combined into a single commend.
-r	Include base class information.

Table 4. Select options of typefinder

As an example, the command shown in Listing 23 yields the output seen in Listing 24.

```
clix typefinder -xip System.String
```

Listing 23. Working with typefinder

Listing 24. Viewing the properties and interfaces of the System.String type

```
class System.String
# Interfaces: 4
interface System.IComparable
interface System.ICloneable
interface System.IConvertible
interface System.Collections.IEnumerable
Properties
Char Chars [Int32] 'get'
Int32 Length 'get'
```

When working with typefinder, be sure to check out the role of the `-d` option (as it will prove helpful when you wish to view types in external assemblies, such as SimpleJScript.exe).

THIS LITTLE GUI IS PLATFORM INDEPENDENT

To wrap up this installment, we will examine what could be the most inciting set of sample applications, the Platform Independent GUI (PIGUI). As mentioned earlier in this series, the SSCLI does not ship with an implementation of System.Windows.Forms (.NET's native GUI toolkit). However, to illustrate what can be done with the CLI, we are provided with a set of samples that leverage the tcl/tk libraries, represented on the Macintosh by two *.dylib files (libtcl8.4.dylib and libtk8.4.dylib). As a friendly reminder, recall that the process of installing tcl/tk was described in the November installment.

The SSCLI offers a handful of samples that leverage the tcl/tk libraries:

- Hello: A simple application illustrating GUI event handling
- PigPad: A text editor application
- Tk: This sample builds the required assembly (sharedsourcecli.tk.dll) that wraps a subset of the tcl/tk libraries

The Tk sample code is very illuminating for a few reasons. First, the code base illustrates how the .NET platform can communicate with the API of the underlying operating system (as well as native libraries) using a specific *attribute* named `DllImportAttribute` (attributes enable aspect-oriented programming [AOP] techniques under .NET, the full details of which will be described at a later time).

On a more practical level, the assembly generated by the Tk sample (sharedsourcecli.tk.dll) is required by all other PIGUI examples. Given this, you must ensure that a copy of sharedsourcecli.tk.dll is placed within the same directory as the executing assembly. Furthermore, you will most likely want to copy the tcl/tk libraries are in the same directory of the pigui

sample application you wish to run. If you installed tcl/tk using Fink Commander, they should be under /sw/lib.

Enabling an X11 Session

To execute the PIGUI sample applications, you must initialize X11 (full screen or rootless), and run these SSCLI samples from an xterm session (as the Macintosh Terminal applet does not initialize the necessary X11 variables). Now, when you install Panther, you have the option to install X11, which it may be launched using the X11 applet located under the Applications/Utilities directory.

If you are not running Panther, you must obtain X11 from cyberspace. While there are numerous distributions of X11 for the Mac, I'd suggest you download and install Apple's office X11 implementation (consult <http://www.apple.com> for further details). For the remainder of this article, I will assume you are at least somewhat comfortable manipulating X11.

Playing with the PIGUIs

By default, Apple's X11 distribution makes use of the Quartz windows manager (quartz-wm), which can be verified by investigating your machine's .xinitrc file. Assuming you have activated X11, enable the SSCLI from an xterm window (via DoSScli) and navigate to the pigpad sample application located under the /sscli/build/v1.ppcfstchk.rotor/samples/pigui/pigpad directory. Finally, execute pigpad via clx. **Figure 6** shows PIGUI running under the default Quartz windows manager.

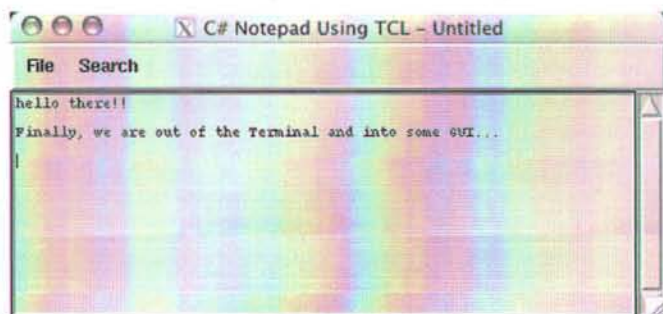


Figure 6. PigPad under the Quartz Window Manager

If you edit your .xinitrc file to load an alternative window manager (which may be obtained via Fink Commander) you would find that pigpad takes on an entirely different look and feel. To prove the point, update .xinitrc to load your window manager of choice (**Listing 24** illustrates the loading of Window Maker).

Listing 24. Loading Window Maker as the X11 Windows Manager

```
# The Apple WM.
# quartz-wm

# Window Maker
exec /sw/bin/wmaker
```

Once you have updated and saved .xinitrc, close down and then restart X11. Now follow the previous steps to execute the pigpad sample application. **Figure 7** shows the new look and feel.

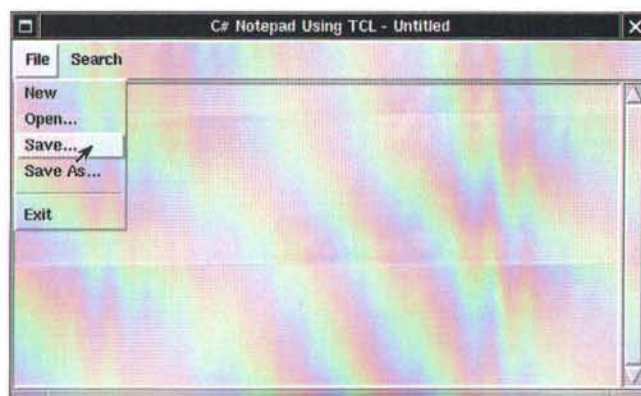


Figure 7. PigPad under Window Maker

Very cool! If you are interested in digging into the "who, what, where and why" of how pigui does it's magic, your first step is to examine the documentation of the Tk wrapper (/sscli/samples/pigui/tk/tk_wrapper.html). Once you have a general sense of the exposed functionality, you should be able to alter the code base of pigpad (or pighello) to your liking.

WRAP UP

This installment rounded out your understanding of the SSCLI by showcasing a number of sample applications and managed languages. We began by exploring the JScript.NET programming language and learned how jsc can be used to build .NET assemblies using a scripting like syntax. We also took a very brief look at the cLisp and myC compilers, and examined a set of sample applications that allow you to explore the format of a given assembly in a more lightweight manner than ildasm. Finally, you had a chance to see how tcl/tk was used to build a platform independent GUI under the SSCLI.

The next issue marks a conceptual shift in this series. While these first issues concentrated on the overall nature of the .NET platform and the role of the SSCLI, the next several articles will pound of the syntax and semantics of C#, and see how this programming language expresses the type system (classes, interfaces, structures, enumerations and delegates) of .NET.

By Tom Djajadiningrat, Designed Intelligence Group, Eindhoven University of Technology

PIC Microcontroller Programming on MacOSX

Using a VOTI Wisp628 with JAL and XWisp

ABSTRACT

This article explains how to program a Microchip PIC microcontroller on a Macintosh running OSX. The programmer hardware that is used is a Wisp628 by Van Ooijen Technische Informatica. The software consists of two parts, JAL and XWisp. JAL is an open source, high-level language that is used to generate a PIC compatible hex file. XWisp is Python-based, open source software that is used to upload the hex file from the Macintosh host via the Wisp628 programmer board to the PIC microcontroller. The article explains how to compile JAL for MacOSX, how to create a connection from the Mac via the Wisp628 programmer board to the target circuit containing the PIC, how to upload a hex file using XWisp, and how to create a simple LED flashing program.

INTRODUCTION

Software for electronic engineering purposes has been far from a stronghold for the Macintosh. Especially affordable, electronic hobbyist level software that is compatible with a Mac has traditionally been hard to find. Luckily, with the arrival of OSX this has changed somewhat. Due to OSX's UNIX underpinnings it is possible to recompile many open source software packages to run on a Mac.

In this article we show how to use two pieces of open source software for microcontroller programming on an OSX Mac. One we will recompile using the GCC compiler, the other we can run using Python. Both the GCC compiler and Python are standard components of the Apple Developer Tools for OSX. Don't worry if you have never done any of this before: this article is meant as an absolute beginner's guide. We'll take it nice and slowly, guiding you through the whole process, one step at a time.

REQUIRED HARDWARE AND SOFTWARE

This is what you need to get PIC programming on a Mac.

Hardware

- Macintosh running OSX.2 or OSX.3
- a USB-serial adapter. There are several types on the market. We tested two:
 - Keyspan USB High Speed Serial Adapter USA-19QW (**Figure 1**)
costs: approx. 40-50 USD
info: www.keyspan.com
 - GMUS-03 USB Serial Adapter (**Figure 2**)
costs: approx. 22 USD
available from: www.voti.nl
- XWisp 628 programmer board (**Figure 3**)
available from: www.voti.nl.
costs: assembled approx. 35 USD. As a kit: approx. 24 USD.
- DB9 'straight-thru' male-female serial cable (if your physical workspace is close to your USB port you may be able to do without it and connect your USB-serial adapter straight to the XWisp 628 programmer board)
- Microchip PIC-16f877 microcontroller or its slightly cheaper successor, the PIC-16f877A. These are popular PICs with high specs.
available from: most electronics stores, including www.voti.nl.
costs: approx. 10 USD for the 16f877 or 8.50 USD for the 16f877A
- Electronics breadboard
available from: most electronics stores, including www.voti.nl
costs: from approx. 11 USD

Recently, **Tom's** wife gave him a brand-new, 3.5kg, non-Apple laptop. Admittedly, he now uses his Ti-Book much less as his new laptop is in every way more intelligent and fun and appears to become even more so by the day. Its only disadvantage may be that it also grows even heavier every day. You can flame him on his Apple infidelity at j.p.djajadiningrat@tue.nl



Figure 1: Keyspan USB High Speed Serial Adapter USA-19QW



Figure 2: GMUS-03 USB-serial adapter

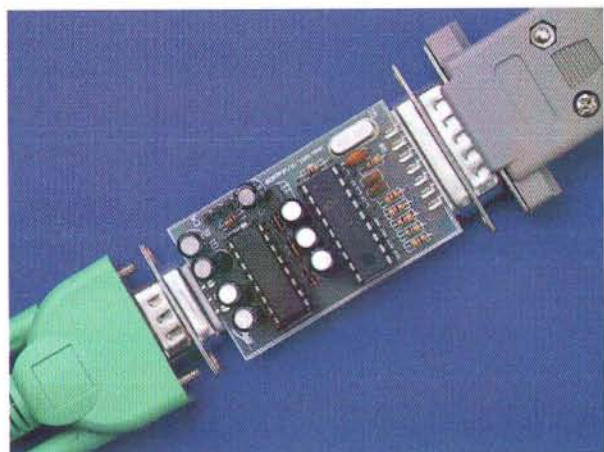


Figure 3: The XWisp 628 programmer board by VOTI. The left-hand side is connected to the USB-serial Adapter through a straight-thru DB9 male-female serial cable, the right-hand side accepts a DB15 connector leading to the target circuit.

SOFTWARE LOCALIZATION MADE

easy

POWERGLOT FEATURE HIGHLIGHTS

- LOCALIZE CLASSIC, CARBON™, COCOA® AND PALM OS™ APPS
- LEVERAGE EXISTING TRANSLATIONS
- AUTOMATE WITH APPLESCRIPT®
- IMPORT /EXPORT TRANSLATION MEMORIES

www.powerglot.com
browse, translate, click, done.

PowerGlot Software - Localization tools for Mac® OS
www.powerglot.com
info@powerglot.com

Mac OS, Carbon, Cocoa and AppleScript are trademarks of Apple Computer, Inc.
Palm OS is a trademark of Palm, Inc.

Software

- Apple Developer Tools
for MacOSX.2: December 2002 Developer Tools + August 2003 gcc Updater
for MacOSX.3: Xcode Tools v1.1
free download from: <http://developer.apple.com/tools/download>
- Keyspan drivers.
Make sure that you have a version that is compatible with your version of OSX.
free download from: <http://www.keyspan.com/downloads/macosx/>
- JAL, latest distribution. We used 0.4.59.
free download from: <https://sourceforge.net/projects/jal/>
- XWisp
free download from: http://www.voti.nl/xwisp/xwisp_src.tar.gz

The total costs add up to around 77-105 USD excluding shipping and handling, depending on your choices. As you may have noted, the GMUS-03 USB-serial adapter is considerably cheaper than the Keyspan. GMUS-03 Drivers for every flavour of OSX are available and the thing works admirably for what we are doing in this article. However, whilst Keyspan have certified their adapter with a great many serial devices, with the GMUS-03 your mileage may vary.

You can save some costs by buying the WISP628 programmer in kit-form, rather than ready-assembled, and by going for a cheaper model PIC.

JAL

We tackle JAL in three parts. Firstly, we prepare JAL for OSX by making a minor change to the code, then we recompile JAL, and, finally, we try out our newly created JAL executable.

Preparing JAL's source code for OSX

Assuming that you have downloaded all the required software components, we start with compiling JAL for Mac OSX. To do this you have to work with Terminal. In the Finder, start up Terminal which is located in the folder /Applications/Utilities. Now we have to navigate to the JAL folder. I dropped mine in Applications. The easiest way to change directory in Terminal is to type `cd`, followed by a space, and then drag the folder you want to navigate to from the Finder onto the Terminal window. Terminal will then add the correct, absolute path. In my case, the Terminal window says:

```
J-P-Djajadiningrats-Computer:~ Tom$ cd /Applications/jal-0.4.59
```

In your case, Terminal may say something different, depending on the version of the JAL distribution and on where you put it. Type **ENTER** to change to this directory.

From now on, we use a dollar sign to represent the Terminal prompt. For all clarity, you only need to type what follows the dollar sign, not the dollar sign itself. Also, this was the first and last time we mentioned that you need to type **ENTER** after each Terminal command.

Do a listing of the Jal-0.4.59 directory by typing:

```
$ls -F
```

This should give you the contents of the JAL directory as shown in the screenshot in **Figure 4**. The operand `-F` makes that the listing displays directories followed by a slash (`/`) and executables by an asterisk (`*`).

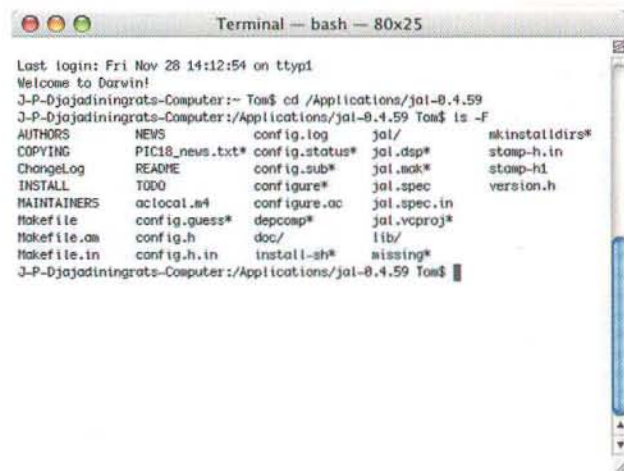


Figure 4: Navigating to and listing the jal-0.4.59 folder

Within the Jal-0.4.59 folder, we now navigate to the source code directory which is simply called `jal`:

```
$cd jal
```

To see a listing of the source code in the `jal` directory you can again type:

```
$ls -F
```

Before we can start compiling we have to make a small change to the file `stdhdr.h` (note that if you have a newer release than 0.4.59, this change may not be necessary anymore). Type:

```
$open stdhdr.h
```

Depending on what version of the Apple Developer Tools you are using, `stdhdr.h` opens in either ProjectBuilder or Xcode. Look for the piece of code in **Listing 1a**.

Listing 1a: stdhdr.h

```
#ifndef HAVE_MALLOC_H                                     #ifdef HAVE_MALLOC_H
#ifdef HAVE_MALLOC_H
#include <malloc.h>
#else
void *malloc(int);
#endif
#endif
```

Comment out these lines, C style, by adding slash-asterisk (`/*`) in front and asterisk-slash (`*/`) so that the block of code looks like **Listing 1b**.

Listing 1b: stdhdr.h

```
#ifndef HAVE_MALLOC_H
/*
#ifdef HAVE_MALLOC_H
#include <malloc.h>
#else
void *malloc(int);
#endif
*/
#endif
```

Don't worry too much about why you need to make these changes. Basically, on OSX we already get access to malloc by including stdlib.h, and therefore including malloc.h gives 'already defined' errors. Close the file in ProjectBuilder/Xcode and save your changes. Now we are ready to start compiling.

Compiling JAL

Switch back to Terminal and change to the directory jal-0.4.59 which means we have to go one directory level upwards. Type:

```
$cd ..
```

The next step is to configure compilation for your Mac by launching the UNIX executable configure:

```
./configure
```

Running it takes a little while, during which you see a whole bunch of checks rolling by. Finally, Terminal should print the message in **Listing 2**:

Listing 2: ./configure

```
jal-0.4.59 is now configured for
Closing remarks during compilation
Build:      powerpc-apple-darwin7.0.0
Host:      powerpc-apple-darwin7.0.0
Source directory:
Installation prefix: /usr/local
C compiler: gcc -g -O2
```

Now type:

```
$make
```

This takes longer, as gcc compiles JAL. To complete the installation process type:

```
$sudo make install
```

Note the sudo (superuser do) command. It allows an administrator—which presumably you are when you are on your own Mac—to run commands as superuser. Basically, becoming superuser upgrades your privileges so that you may access files and directories which normally are protected against (accidental) misuse. In this case, we want to install into the /usr/local/bin directory which is a protected directory. After entering this command, Terminal therefore asks you for your password. Once you have given it, Terminal responds as per **Listing 3**.

Listing 3: \$sudo make install

```
Response to $sudo make install
/bin/sh ../mkinstalldirs /usr/local/bin
/usr/bin/install -c jal /usr/local/bin/jal
make[1]: Nothing to be done for `install-data-am'.
```

As you can see, the JAL executable ends up in the directory /usr/local/bin. To convince ourselves, change to the installation directory and do a listing:

```
$cd /usr/local/bin
$ls -F
```

And there you have it: jal followed by an asterisk, indicating that this is an executable. But there is more to the Jal installation than just the executable. There are also several include files. Let's see whether we can find them.

Change to the parent directory /usr/local and do a listing:

```
$cd ..
$ls -F
```

In it you see a directory named share. Change to it and do a listing:

```
$cd share
ls -F
```

There you find a directory called jal. Change to it and do a listing:

```
$cd jal
$ls -F
```

Finally, there is the lib directory. In this directory you find a couple of dozen of include files:

```
$cd lib
$ls -F
```

I hope this gives you a feel for where your Jal installation ends up.

Taking JAL for a spin

Create a directory somewhere convenient for your JAL source files. Mine is called src and lives on a partition called Data. Now we create a JAL source file. First, using a text editor of your choosing (I used XCode), create a new empty file and type in the code in **Listing 4**. Don't worry about the exact meaning of the code for the moment. If you are curious, later on we will use this JAL program to flash an LED connected to pin a0 of our PIC:

Listing 4: ledflash.jal

```
The full ledflash.jal source file
include 16f877_20
include jlib
disable_a_d_functions

pin_a0_direction = output

forever loop
    pin_a0 = high
    delay_10mS(255)
```



```
pin_a0 = low
delay_10mS(50)
end loop
```

Save the file into your source file directory under the name `ledflash.jal`. The filename itself is up to you, but it is critical that you use the `.jal` extension.

Change to your source file directory by typing `cd`, followed by a space and then dragging the directory from the Finder onto your Terminal window:

```
$cd /Volumes/Data/src
```

Do a sanity check verifying that the file `ledflash.jal` is actually in there:

```
$ls
```

And now, for the moment suprême. Let's try compiling `ledflash.jal` using our `jal` executable:

```
$/usr/local/bin/jal ledflash.jal
```

As `ledflash.jal` is compiled, you see the lines flash by and, finally, Terminal responds as in **Listing 5**:

Listing 5: `$/usr/local/bin/jal ledflash.jal`

```
JAL compiling ledflash.jal
jal 0.4.59 (GCC 3.3)
input      files:12 lines:2244 chars:58707 (2952
lines/second)
compilation nodes:12695 stack:53Kb heap:3928Kb seconds:0.760
output     code:101 file:14 stack:2
OK
```

Have a look what is in our `src` directory now:

```
$ls
```

There you have it: in addition to a `.jal` file, there are now an assembler file (`ledflash.asm`) and a hex file (`ledflash.hex`). Drag them onto your favourite text editor to see their content:

The hex file is very compact as shown in **Listing 6**.

Listing 6: `ledflash.hex`

The `ledflash.hex` file resulting from compiling `ledflash.jal` with JAL

```
:020000040000FA
:020000000428D2
:08000800FF30A100FF30A2004F
:10001000FF30A300FF30A4000F30A5008A110A12A0
:100020004E2021108A110A125C2026148A110A120D
:100030005620FF308A110A122B2026108A110A122C
:10004000562032308A110A122B208A110A121528E2
:100050008A110A122828A7002708A8006430A900DE
:1000600001308A110A123428AA00FF30AB00290897
:10007000AD002808AC0077308A110A122B0703184C
:100080003E288A110A12AC0B3B288A110A12AD0BCA
:1000900039288A110A12AA0B372808008A110A1275
:1000A000622007309F008A110A1265288A110A12FD
:1000B00059282608850008008A110A125F2821089D
:1000C0006500080083160313080083120313080059
:02400E00723FFF
:00000001FF
```

The assembler file is something you are supposed to never need. The gurus use it to debug the compiler and to learn how assembler works, ie. to figure out what the PIC really does for $A = B + C$.

So far so good. We can write a simple `.jal` file and compile it using an OSX native JAL executable, resulting in a hex file suitable for a PIC microcontroller. Now how do we actually upload this hex file from the Mac to the PIC?

UPLOADING THE HEX FILE

First, we need to get our physical, serial connection between the Mac, the Wisp628 and the PIC microcontroller in order, then we look into how to use the XWisp software to upload our hex file to the PIC.

Serial connection

Plug your USB adapter into the USB port of your Mac. Assuming that you are using a Keyspan, you can do a check by running the Keyspan Serial Assistant which lives in your Applications folder. The Serial Assistant should recognize the adapter (**Figure 5**).

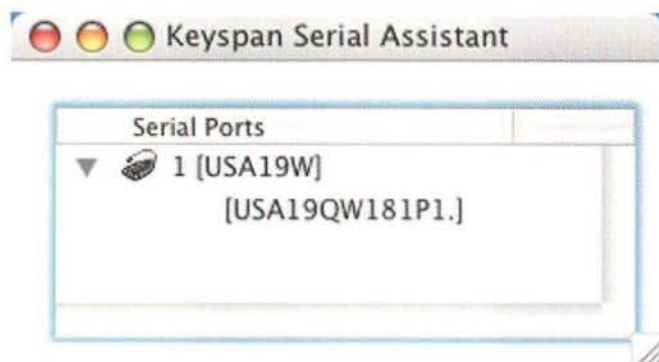


Figure 5: The Keyspan Serial Assistant acknowledges the adapter

Depending on the cable length you need between your USB port and your physical workspace, you can connect the Wisp628 board directly to the USB adapter or connect it using a 'straight-thru' DB9 serial cable,

With the Wisp628 comes a cable which has a DB15 female plug on one end and which has bare wires on the other end. Plug the DB15 female plug onto the Wisp 628 board.

Plug the 16f877 microcontroller into a breadboard

The necessary circuitry consists of two parts: a regulated 5V power supply (**Figure 6**) and a target circuit including the PIC (**Figure 7**). As these circuit diagrams may turn out to be hard to read when scaled to the width of a single MacTech column, the code archive includes PDF versions of both.

16 ~~15~~ years and running.

Ever since we built our first Ethernet adapter in 1987, we have been devoted to making networking faster, easier, and safer.

Now, our comprehensive line of Gigabit Ethernet solutions makes high-speed networking simpler than ever – at prices below \$20 per port.

Mac OS. Windows. Linux. The choice is clear.

Asanté Technologies.

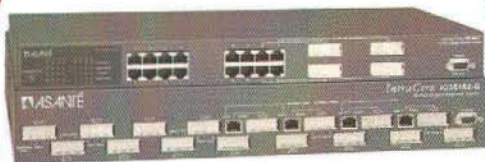
FriendlyNET® GX5-W 10/100/1000 Gigabit Ethernet Smart Switches

NEW!



Deliver Gigabit Ethernet to the desktop with comprehensive management via any web browser!

IntraCore® Multi-service Gigabit Ethernet backbone switches



Uncompromised flexibility in any environment!

GigaNIX™ – the first Gigabit Ethernet adapter compatible with Mac OS, Windows, and Linux

Gigabit speeds over existing copper wiring!



"FriendlyNET GX5-424W ... its price tag is nice and short"
Processor.com

"Asanté GigaNIX ... an impressive set of features at a great price."
Networks Today

"Asanté IntraCore ... well suited for handling multi-media traffic including VoIP and video, even in a mixed computing environment."
PC Magazine

 **ASANTÉ**

1-800-662-9686
www.asante.com



© 2004 Asanté Technologies, Inc. Asanté, FriendlyNET, and IntraCore are registered trademarks and GigaNIX is a trademark of Asanté Technologies. All other trademarks are property of their respective owners. All rights reserved.

Have an older Mac?
See asantestore.com
for all of your
networking needs.

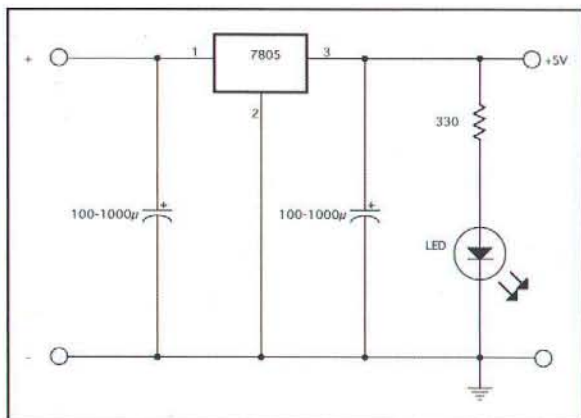


Figure 6: A regulated 5V power supply

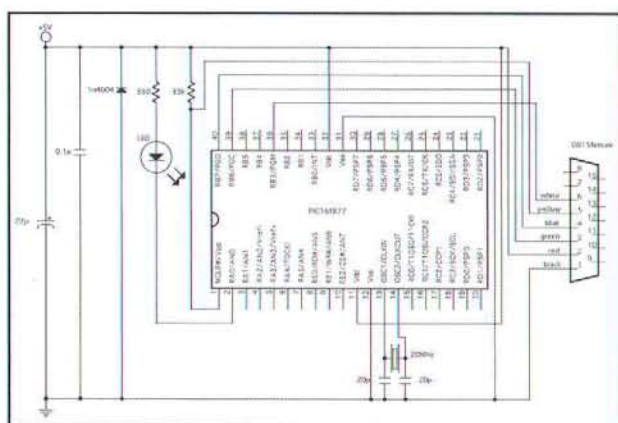


Figure 7: The target circuit with the PIC

You can build a stable 5V supply starting from a 9V battery or from a 9V DC power supply. I would recommend that you use a 9V DC power supply rather than a battery since the voltage regulator tends to eat through 9V batteries rather quickly.

Components for regulated 5V power supply:

- 1x 9V DC power supply or a 9V battery with a clip-on adapter
- 1x 7805 voltage regulator
- 1x green LED
- 2x 330Ω resistor
- 2x 100-1000µF capacitor
- The resistor and the LED are not strictly necessary but do make the final check very easy: the LED should light up.
- Components for the target circuit with PIC:
 - 1x 330Ω resistor
 - 1x 33kΩ resistor
 - 1x red LED
 - 1x 20MHz crystal
 - 2x 20pF capacitors
 - 1x 1N4004 diode

- 1x 22µF capacitor
- 1x 0.1µF capacitor

The 1N4004 is a 'fool's diode': if you accidentally reverse the polarity of your power supply it will prevent damage to your PIC. Of course, the diode has its limits. If you happen to use a supply that is capable of delivering high currents (>1A), such as an old PC power supply, the diode will blow and the PIC will be damaged after all.

All that is left now, is to connect the Wisp628 board to the PIC in the breadboard and build a small test circuit around the PIC. This is detailed for various PIC microcontrollers on http://www.voti.nl/wisp628/n_index.html. Here we focus on the connections for a 16F877 microcontroller (**Figure 8**).

DB15 pin#	DB15 cable strand colour	PIC pin#	PIC pin name
1	black	12, 31	Vss (Gnd)
2	red	11, 32	Vdd (+5V)
5	yellow	1	MCLR#/Vpp
6	white	36	RB3/PGM
3	green	39	RB6/PGC
4	blue	40	RB7/PGD

Figure 8: Connections between the DB15 connector of the XWisp 628 and the target circuit with the PIC

Figure 9 shows how things were wired on my breadboard. Be aware that some breadboards, including this one, have breaks in the middle of the power rails which you need to bridge if you want to use both the left and the right half. Again, a high-res JPEG picture is included in the code archive.

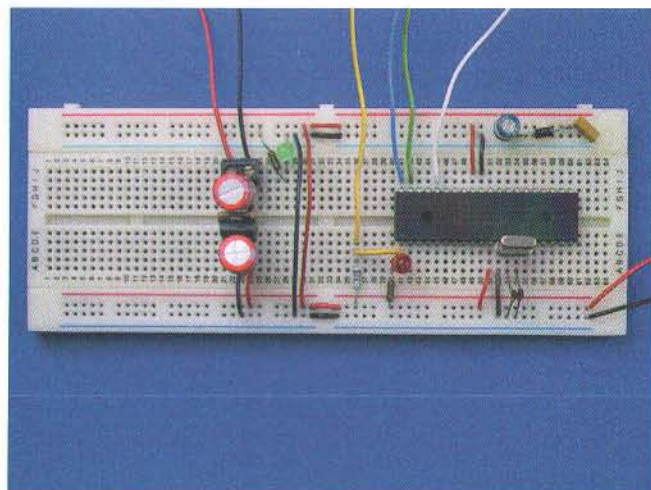
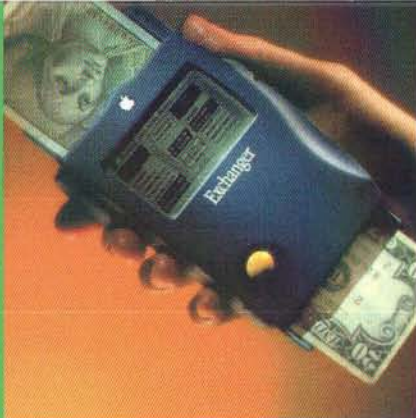


Figure 9: The 5V power supply and the target circuit with the PIC. Everything to the left of the wire bridges in the power rails forms part of the 5V power supply, everything to the right forms part of the target circuit with the PIC. The red and black lead coming in from the top-left are connected to the 9V battery/power-supply. All the other flying leads go to the Wisp 628.

32
< 1 year
2 years >
\$64



INTERVIEWS

TAPPING INTO THE WORLD OF CELEBRITIES AND THEIR MACS, ONLY MACDIRECTORY OFFERS EXCLUSIVE INTERVIEWS. GET A CLOSE AND PERSONAL VIEW FROM SARAH JESSICA PARKER, STING, STEVE JOBS, MADONNA, HARRY CONNICK JR., GEORGE LUCAS, JENNIFER JASON LEIGH, STEVE WOZ AND OTHER LEADERS IN THE MAC COMMUNITY.



FEATURES

DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS & OUR TECHNICAL EXPERT TEAM OFFER THEIR OWN PERSONAL INTERPRETATION OF THINGS THAT ONLY THE MAC SYSTEM CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, INTERVIEWS, NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH BUYERS' GUIDE INCLUDING OVER 5,000 MAC PRODUCTS AND SERVICES.



CULTURE

MACDIRECTORY TAKES YOU TO THE WILDEST CORNERS OF THE WORLD AND UNCOVERS HOW MACINTOSH COMPUTERS ARE BEING USED BY OTHER CULTURES. TRAVEL TO JAPAN, AUSTRALIA, GERMANY, BRAZIL, INDIA, RUSSIA & LEARN MORE ABOUT APPLE'S CULTURAL IMPACT AROUND THE GLOBE.

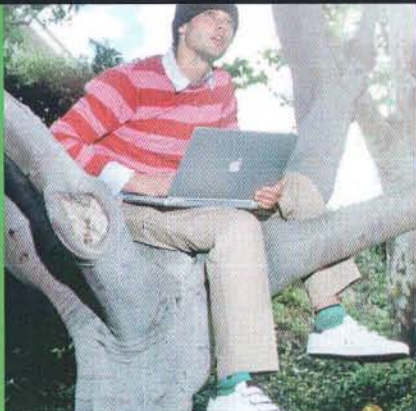
MacDirectory

BEYOND ANY MACINTOSH MAGAZINE. SUBSCRIBE.

< Subscribe

www.macdirectory.com/mw.html

SEND CHECK OR MONEY ORDER TO:
MACDIRECTORY SUB DEPT.
326 A STREET, 2C
SOUTH BOSTON, MA 02110



REVIEWS

FIND OUT ALL YOU NEED TO KNOW ABOUT THE LATEST MAC PRODUCTS INCLUDING THE HOTTEST MAC OS SOFTWARE AND HARDWARE.



WIN!

SUBSCRIBE TO MACDIRECTORY AND YOU WILL AUTOMATICALLY ENTER OUR SWEEPSTAKES FOR A CHANCE TO WIN A NEW TITANIUM!

Using XWisp

Now we can work on the software side of uploading the ledflash.hex file to the target circuit. For this we use the Python-based XWisp. Before we can issue the upload command we need to do three things. Firstly, we make a small modification to one XWisp source file to enable XWisp to run on OSX (Just as with the Jal source, you may find that in the latest release of XWisp, this modification may not be necessary anymore). Secondly, we copy the XWisp folder to /usr/bin. And, finally, we find the name of the serial port provided by the USB-serial adapter and its driver software.

Modifying XWisp

In the xwisp_src Folder that you downloaded you find six files. Drop the file xwisp.py on ProjectBuilder or Xcode to open it. Use the Find command to look for CMD_PORT. In this definition, look for the line of code saying:

```
Name = self.Get_arg( Uppercase = 0 )
```

After this line, we need to add two lines:

```
self.Port = Name # these two lines are added
return # to use the port name 'as is'
```

Please note that Python code is indentation sensitive. Indents are to Python what curly brackets are to C. You need to make sure that the two added lines are on the same indentation level as the line

```
if Name == None:
```

Just to make sure, the finished version of the CMD_PORT definition is shown in the screenshot in **Figure 10**.

```
def CMD_PORT( self, Name = None ):
    self.Close_Bus()
    if Name == None:
        Name = self.Get_Arg( Uppercase = 0 )
    self.Port = Name # these two lines are added
    return # to use the port name 'as is'
    if (Name + '')[0:3].upper() == 'COM':
        if (int((Name + '')[3:]) > 9):
            self.Port = '\\\\.\\" + Name
        else:
            self.Port = Name
    else:
        N = self.Int_Value( Name )
        if N > 32:
            self.Active_Baudrate = N
        else:
            self.Port = N
```

Figure 10: a screenshot of the CMD_Port module after the necessary changes

Copying XWisp

Rename the folder named xwisp_src Folder to just xwisp. Now we need to copy the folder to /usr/bin. Unfortunately, we cannot simply use the Finder as the folder /usr is hidden. Therefore we again turn to Terminal. Type `sudo cp -R`, followed by a space, then drag our xwisp folder onto Terminal like we did before, and finally type `/usr/local`. In my case, the command looked like:

```
$sudo cp -R /Users/Tom/Desktop/xwisp /usr/local
```

This copied the xwisp folder with all its content. Now do a listing of the directory /usr/local.

```
$ls -F /usr/local
```

In the listing you should find the directory xwisp. If you wish to convince yourself that we have not only copied the folder but its contents too, do a listing of the contents of the xwisp directory:

```
$ls /usr/local/xwisp
```

Finding out the name of the serial port

In terminal, type:

```
$ls /dev
```

If you have the Keyspan adapter, look for a serial port named something like `tty.USA19QW181P1.1`. The name may differ with the exact type of Keyspan adapter you have got. If you use a GMUS-03, look for a port named something like `tty.usbserial0`. Note down the name somewhere as we will need it in a moment.

We are nearing yet another moment suprême. To run XWisp you need Python which luckily forms part of the Apple Developer Tools.

Change to the directory where you keep your .jal, .asm and .hex files. In my case:

```
$cd /Volumes/Data/src
```

OK, ready? We have xwisp.py living in /usr/local/xwisp, ledflash.hex living in our working directory and a serial port living in /dev. Now, on a single line, type this if you have the Keyspan adapter:

```
$python /usr/local/xwisp/xwisp.py port
/dev/tty.USA19QW181P1.1 go ledflash.hex
```

Or this if you use the GMUS-03 adapter:

```
$python /usr/local/xwisp/xwisp.py port /dev/tty.usbserial0 go
ledflash.hex
```


If you are (very) lucky, the LED connected to pin a0 of the PIC microcontroller starts flashing. No joy? No need to panic, yet. With a setup with this many components, it is unlikely that you get things running the first time round. Which brings us to...

TROUBLE SHOOTING

Stuck?

Here are some things to check:

Do you manage to run XWisp?

When you try to run XWisp, Terminal should respond with printing some lines, the first of which reads:

```
XWisp 1.08, command line mode
```

If it does not, make sure that XWisp really lives where you think it lives.

Are you really reaching that serial port?

Usually, you can figure out from the error message in Terminal if you do not address the serial port correctly (If you use the Keyspan adapter you can look at its LED: it should start flickering during uploading of the hex file). If not, check the physical connection between Mac and USB-serial adapter, check the name of the serial port, and make sure that you have specified the complete path: you really need that `/dev/` in front of the port name.

Does your power supply (still) work?

Check whether the LED in your power supply circuit is on. But even if it is on, the voltage over the PIC may have dropped to unacceptable levels. The 7805 voltage regulator needs about 2V headroom to do its work: to create 5V output it needs at least 7V input. With a 9V battery you can quickly drop below this 7V level. If you find that you are going through your 9V battery a bit too quickly, you may want to replace it by a 9-12V power adapter. A low supply voltage may lead to all kinds of weird behaviour: at 4.5V the PIC may run its program still fine, but programming it may have become impossible.

Is your target circuit correctly hooked up?

Check the wires between the Wisp628 and your target circuit. Make sure that you really have +5V on pins 11 and 32 and ground on pins 12 and 31.

And finally... is your LED the right way round?

Well, I'm sure you wouldn't be the first... The flat side or shorter lead is the cathode and should be connected to pin a0. The other side, the anode, is connected to +5V. When you are in doubt whether the LED is connected correctly, just disconnect the PIC-side of the LED and connect that side to ground (0 Volt). The LED should light up. If it does not, it is reversed (or there is no power).

Still stuck?

On www.voti.nl you can find some highly detailed pages to help you trouble shoot.

Desperate?

There is a great Yahoo Group for Jal users:

<http://groups.yahoo.com/group/jallist/>

Strictly speaking this group is for Jal questions only, but you would not be the first to start a discussion about the Wisp 628.

CONCLUSIONS

So there you have it: programming a PIC microcontroller on a Mac. It takes a little time and perseverance to set it all up. But it works, have fun!

ACKNOWLEDGEMENTS

I gratefully acknowledge my colleagues Peter Peters and Joep Frens at the Designed Intelligence Group of Eindhoven University of Technology for checking the manuscript. Many thanks also to Wouter van Ooijen of Van Ooijen Technische Informatica to get all of this running, and to Daniel Saakes of the ID-StudioLab of Delft University of Technology for his Python on Mac advice.

REFERENCES

Microchip are the makers of the PICmicro microcontrollers series. Here you can find datasheets for all model PICs.

<http://www.microchip.com>

VOTI, Van Ooijen Technische Informatica, are the makers of the Wisp628 hardware, the XWisp software and the originators of JAL. Here you can find detailed instructions on all their products as well as a web shop.

<http://www.voti.nl>

Keyspan are the makers of a USB-serial DB9 adapter.
<http://www.keyspace.com>

A MacOSX 10.3 Panther compatible driver for the GMUS can be found here (drivers for older versions of OSX are provided on the installation CD).
<http://www.ramelectronics.net/download/BF-810/OSX/>

The Yahoo group 'jallist' is a forum for JAL users.
<http://groups.yahoo.com/group/jallist/>

Apple provides ProjectBuilder and Xcode as free downloads: <http://developer.apple.com/tools/download/>

On source forge the open source community further develops Jal:
http://sourceforge.net/project/showfiles.php?group_id=71552

And finally, here you can find lots of information on PICmicro: <http://www.piclist.com>



By David Linker

NoCode Browser

Using the Apple's Web Kit SDK to make a web browser

WANT SOME RAPID DEVELOPMENT?

One of the advantages that is mentioned about the Cocoa development environment is the well-developed frameworks that are provided, giving impressive functionality without much effort. A recent addition is Web Kit, which is the framework which provides a full suite of components required for web browsing, and are the basis of the Safari web browser from Apple. To illustrate the power of Web Kit, Apple explains in the documentation how to make a browser with only one line of code. Some folks have expanded on this explanation, and in the discussions that ensued, others pointed out that it is possible to write a browser using Web Kit with NO lines of code. This article will explain how, and discuss some of the capabilities of Web Kit as well.

FAST LANE

In this age of 24/7, just-in-time and accelerated learning, some of you may want to get right to the point, while others want more details. To satisfy the more experienced readers, or the less patient, here is the executive summary of the main points necessary:

- Install Project Builder, December 2002, if not already installed
- Install Safari, if not already installed
- Install Web Kit
- Make a new Cocoa Application project in Project Builder
- Add the Web Kit Framework to the project
- Open up the MainMenu.nib file in Interface Builder
- Drag the Webview header file to Interface Builder
- Add a Customview to the main window
- Change its type to Webview
- Add a text box to the main window
- Connect the target output of the text box to `takeStringURLFrom` in the Webview

Build the project
Type a URL in the text box and hit Enter
Browse away!

The rest of this article fills in the details, and add a few niceties along the way.

ORIGINS AND BACKGROUND

Apple built the technology used in its new browser, Safari, on existing open source projects. KDE is an open source "desktop" environment for unix. Part of KDE is a set of tools for rendering HTML, called KHTML, and a set of tools called KJS, which assist with scripting. These two components were used by Apple to develop the core classes that are used in Safari. The framework incorporating these classes which they developed for Safari is called Web Kit, and Apple released the interface to Web Kit during the WWDC in June of this year.

Web Kit provides an amazing amount of functionality with little effort. It supports HTML, DOM, SSL, Javascript, stylesheets, embedding Java applets, and a "history" of recent sites. What this means in practical terms is that it is possible to incorporate all of this functionality in your applications with minimal effort.

Apple provided a very terse description of how to do this on the pages describing the use of Web Kit at:

<http://developer.apple.com/documentation/Cocoa/Conceptual/DisplayWebContent/index.html>

If you choose "Simple Browsing", there is a short description of how you can create a browser with one line of code, which is as follows:

```
[[webView mainFrame] loadRequest:[NSURLRequest
requestWithURL:[NSURL URLWithString:urlText]]];
```

Of course, there is a fair amount of "wrapper" that has to go around this to incorporate it in a program, including making a header file, class file, and all of the connections in the interface.

David is a lover of Mac OS X, because the rich development environment and frameworks allow his inherent laziness to blossom. You can reach him at dtlinker@mac.com.

ThinkfreeOffice

COMPATIBLE WITH MICROSOFT WORD, EXCEL AND POWERPOINT

WORDPROCESSOR • SPREADSHEET • PRESENTATION GRAPHICS

The Affordable Office Alternative!

Three High-Performance Applications

Thinkfree Write

Thinkfree Write is a powerful word processing application that enables you to create rich, professional quality documents and Web pages. You can insert tables, images, and clipart, or even apply custom layouts to your document...then effortlessly proofread your work with the easy-to-use spelling and auto-correction features.

Thinkfree Calc

Thinkfree Calc is a full-featured, easy-to-use spreadsheet application that can easily tackle the most complex analytical tasks with over 40 charts and 300 function capabilities. Thinkfree Calc opens, edits, and saves directly into the Microsoft Excel (.xls) format, so users can seamlessly share documents and collaborate with Microsoft Office users.

Thinkfree Show

Thinkfree Show enables you to create high-impact presentations including animation effects, drawings, images, clipart, and other graphic features. Thinkfree Show opens, edits and saves directly into the Microsoft PowerPoint (.ppt) format.

CyberdrivePlus

A free, one-year subscription to CyberdrivePlus is also included. CyberdrivePlus provides you with secure, Internet file storage and free online software upgrades!



"Thinkfree is a best-of-breed program that will exceed your expectations."
— Jeffery Battersby



"Thinkfree Office is the next best thing and then some."
— Deborah Shadovitz



"Thinkfree Office is an impressive attempt to crack the seemingly impenetrable productivity market."
— Chris Ward

amazon.com

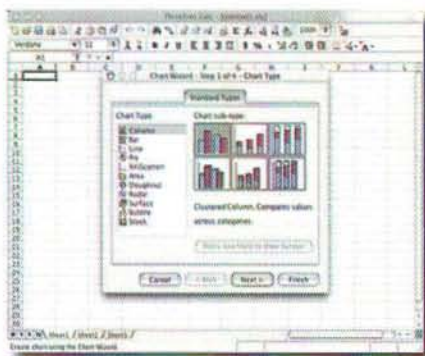
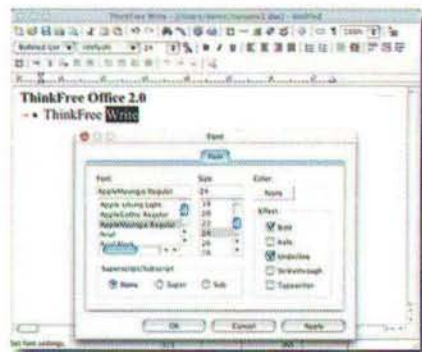
Apple Store



Apple Specialist



ONLY
\$49⁹⁵



NAME THAT TUNE!

Many years ago, was a TV game show called "Name that tune". In it, the contestants would try to name a tune in the fewest notes, challenging each other with "I can name that tune in N notes", where N was a small number, and the person who named the smallest number got the first chance.

I had a flashback of that show when I was reading through a discussion about using Web Kit. The original page, by Martin Simoneau, was an excellent article on Cocoa Dev Central filling in the details on how to make a browser using the line of code provided by Apple, and the Web Kit framework.

<http://cocoadevcentral.com/articles/000077.php>

In the follow-up discussions that were posted, there were comments that the one line of code was unnecessary! This sounded to me like "I can name that tune in no notes!". There were some very brief descriptions of how to do this, and then a link to another page which described how to do this in slightly more detail.

<http://www.livejournal.com/users/foxmagic/238347.html>

The essential ingredient is the fact that there is already a connection called `takeURLFrom`, which will extract the URL from a text field. This makes it possible to create a functional browser without writing any code.

To do this, there are three essential steps that are necessary. The first is that you need to have Safari installed, since that also installs the Web Kit framework. You can find it at: <http://www.apple.com/safari/download/> if you don't have it already. Next, you need to have the developer tools, December 2002 version installed, if you haven't already. To get this, you need to become an ADC developer, but fortunately you can join for free. The site to get this from is <http://connect.apple.com/>. Follow the links Download Software -> Developer Tools to find what you need.

Finally, you need to get the Web Kit SDK. This is also available at the same site, under Download Software -> WWDC 2003. Once you have installed all of the software, you are ready to go.

Start up Project Builder, and choose New Project from the File menu. Pick Cocoa Application from the list, and enter the name `NoCodeBrowser` in the Project Name field. Click on Finish.

Now, we have to add the Web Kit interface to the project. Choose **Add Frameworks** from the Project menu (see **Figure 1**). On the list that comes up, choose `Webkit.framework` and then click **Add** on the next dialog. If you want to be very neat, you can move the `Webkit.framework` to the folder `Other frameworks` under `Frameworks`. Save the project.

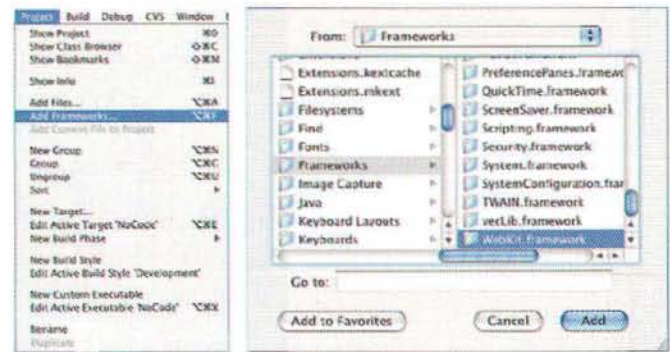


Figure 1 - Select **Add Frameworks** from the Project menu (left), then choose `Webkit.framework` from the dialog that comes up, and click **Add** (right).

We now move to Interface Builder. Click on the little triangle to the left of **Resources** in the project window, and double click on `Mainmenu.nib`. This will open Interface Builder, with a window called "Window". In the window titled `Cocoa-Containers`, click on the second icon from the right on the top, which shows a tabbed window. Drag a `Customview` over to "Window", and drop it in the window. Resize it to fill almost the entire window, with space to put a text box at the top. Click on the second icon from the left in the "Cocoa-Containers" window, and drag a text field to the top of the window, and resize it to make it wider.

Now, arrange the windows so that you can see the window "Mainmenu.nib" in Interface Builder, and the main project window from Project Builder. Open `Webkit.framework`, and the Headers folder inside that (**Figure 2**). At the bottom, there is a file called `WebView.h`. Drag this over to the `Mainmenu.nib` window, and drop it there.

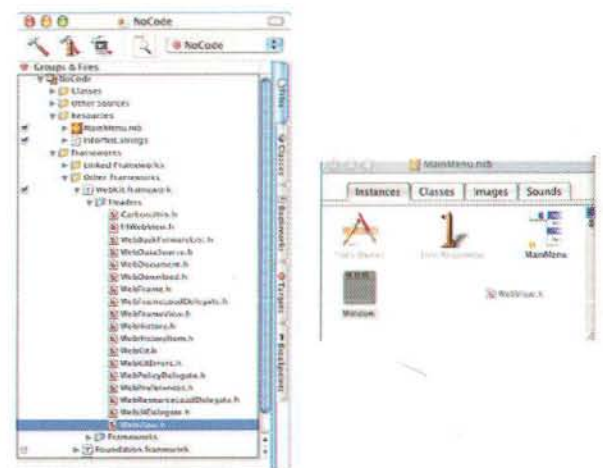


Figure 2 - Choose the `WebView` class from the file list in Project Builder (left), and drag to the `Mainmenu.nib` window in Interface Builder (right).

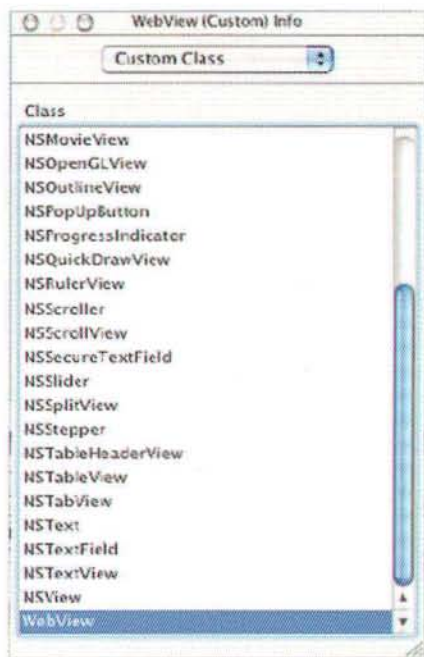


Figure 3 – In the Show Info window, select Custom Class from the pop-up, and then select WebView as the class.

Now, select the CustomView in Interface Builder, and choose Show Info from the Tools menu. On the pop-up menu that

says Attributes, choose Custom Class, and then choose WebView from the list (**Figure 3**). Close the info window. Now, use ctrl-click and drag to make a connection from the text field to the WebView (**Figure 4**). In the window that pops up, make the connection from target to takeURLFrom, and click on connect. Save everything, and then build, using Build and Run from the Build menu.

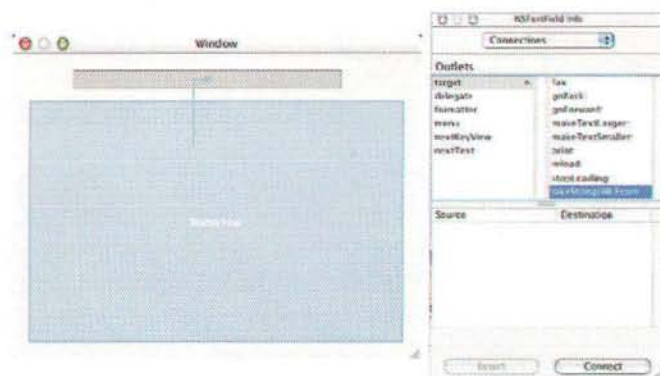


Figure 4 – Connect the text field to WebView and then specify that the connection is to takeStringURLFrom



THE LAW OFFICE OF
BRADLEY M. SNIDERMAN

Need help safeguarding your software?

If you're developing software, you need your valuable work protected with trademark and copyright registration, as well as Non Disclosure Agreements.

Then, when you are ready to sell it, you can protect yourself further with a licensing agreement.

I am an attorney practicing in Intellectual Property, Business Formations, Corporate, Commercial and Contract law.

Please give me a call or an e-mail. Reasonable fees.

23679 Calabasas Rd. #558 • Calabasas, CA 91302
PHONE 818-222-0365 FAX 818-591-1038 EMAIL brad@sniderman.com

You can now type a full URL in the text field, and when you hit enter, the page will load! Note that the URL must begin with "http://". You can then click in links in the loaded page, and those links will load as well. You can navigate to secure pages, load java applets, execute javascript, and lots more. Play around with it!

FEATURE FILL

A number of things are missing but easily added to this first version. First, we can add additional functionality without any more code.

WebView maintains a history of recently visited pages by default. Methods exist in the WebView class to back up a level (`goBack`), go back down a level (`goForward`), reload a page (`reload`), or stop loading a page (`stopLoading`). All we need to do to implement these is to add a button for each function, and connect them to the WebView class and the appropriate method. We can then add appropriate text or icon to each button.

Another problem is that the WebView does not change its size when we resize the window. This is easy to fix. Click on the WebView in Interface Builder, and then choose **Show Info** from the **Tools** menu. From the pop-up, choose **Size** (Figure 5). The box indicates the WebView object, and the straight lines indicate a fixed relationship. If the lines are straight within the object, the size will not change with a resize. If the lines outside are straight, the relationship to the containing window will not change. If all of the outside lines are straight, the object will be centered with a resize. If you click on a line, it turns into a "spring", which will allow resizing. You can do the same thing to all of the other object, such as the buttons and the text box, to control their behavior during resizing as well.

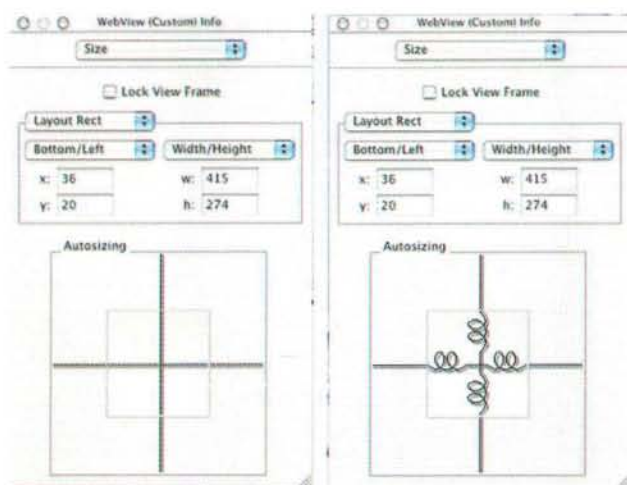


Figure 5 – Click on the interior lines in the box (left) to turn them into "springs" (right), to allow the WebView to resize along with the window.

The final refinement is to change all of the menu items that refer to "NewApplication" to refer to "NoCode Browser".

A picture of the main window in my finished version is in Figure 6.

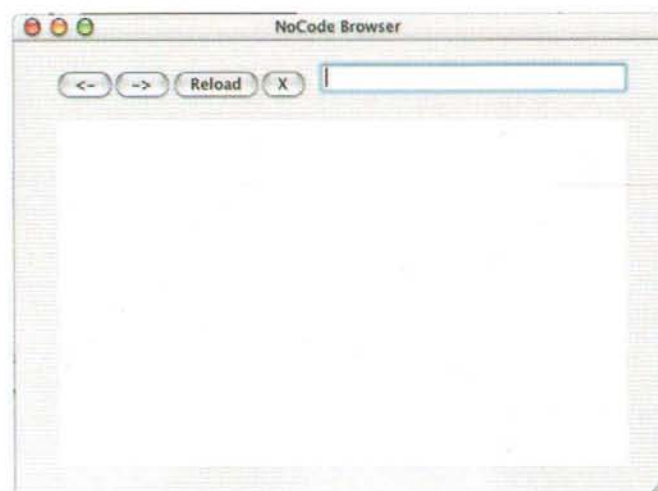


Figure 6 – The final appearance of the main window, after adding back, forward, reload, and stop buttons.

RESOURCES AND ADDITIONS

Although this demonstration is impressive, there are a lot of missing pieces if we were to try to make a complete program.

If you click on a link which should result in opening another window, nothing happens. The same thing goes for email links, download links, and anything other than navigation to another page.

The current version has no error checking, and no error messages. Probably the worst, obvious omission is that if WebKit framework is not loaded, that is, if Safari has not been installed, the program will not work and will probably crash. Methods for dealing with this are explained in the tutorial pages at:

[http://developer.apple.com/documentation/Cocoa/Conceptual/D
isplayWebContent/index.html](http://developer.apple.com/documentation/Cocoa/Conceptual/DisplayWebContent/index.html)

WebKit has a number of hooks to allow changing or enhancing its behavior. The use of these links is also explained at the tutorial pages.

Finally, there is a Web Kit discussion list at:

<http://lists.apple.com/mailman/listinfo/webkitsdk-dev>

In addition to providing impressive functionality that you can use in your programs, WebKit provides a dramatic demonstration of the power of the frameworks available under Mac OS X, allowing you to create a web browser without writing a single line of code. What other platform lets you do that?

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

**Trying to figure out how to handle multiple
compression formats and platforms?**

The Stuffit Engine solves the compression puzzle.



Aladdin's Stuffit Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the Stuffit file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low
as \$99/year

To learn more, visit:
www.stuffit.com/sdk/

Stuffit Engine SDK™ The power of Stuffit in your software.



**Looking for the easiest and fastest
way to build an installer?**

Stuffit InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. Stuffit InstallerMaker makes it simple and effective.

- Stuffit InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with Stuffit InstallerMaker.

Prices start at \$250

To learn more, visit:
[www.stuffit.com/
installermaker/](http://www.stuffit.com/installermaker/)

Stuffit InstallerMaker™ The complete installation solution.™



www.stuffit.com
(831) 761-6200

© 2002 Aladdin Systems, Inc. Stuffit, Stuffit InstallerMaker, and Stuffit Engine SDK are trademarks of Aladdin Systems, Inc. The Aladdin logo is a registered trademark. All other products are trademarks or registered trademarks of their respective holders. All Rights Reserved.

By Michael R. Harvey

Round-up of what we saw this year

ANNOUNCEMENTS, ANNOUNCEMENTS, ANNOUNCEMENTS

This MacWorld Expo saw no earth shattering releases, but a whole pile of new and updated releases from companies across the spectrum of Mac-dom. Some of those announcing, or releasing, new and updated products included Microsoft, with their Office 2004 for Mac, and Timbuk2, with two new cases added to their line of packs and laptop cases. There was also news from Oxford Semiconductor, showing off their OXFW912 FireWire 800 to IDE bridge chipset. LANDesk Software demonstrated their latest LANDesk Management Suite 8, while AITO Technology announced that their host bus adapters fully support both the G5 processor and Panther. What else? Read on.

APPLE

Serious iron for the rest of us

The G5 processor finally made it in to the Xserve. With two 2.0 GHz processors packed in, the new Xserve is capable of ever greater processing performance. You can check out Apple's website to see all the pretty graphs showing off the horsepower of this revision. Not so widely known about the new Xserve G5 is that it now has a hardware RAID built in, which gets you hardware RAID level 5, 150Mbps serial ATA drives, and set rebuilding in the background. Very good news.

Almost more important for network administrators is the updated Xserve RAID. With capacity up to 3.5 Terabytes, fourteen hot swappable drive bays giving you a \$3.14 per gigabyte price point, this updated unit places itself in a very attractive position for any network or server administrator looking to expand their storage. There is also hardware set rebuilding in the background, just like the Xserve G5. This ability in the Xserve RAID is a firmware update, and should be updateable on older Xserve RAID units.

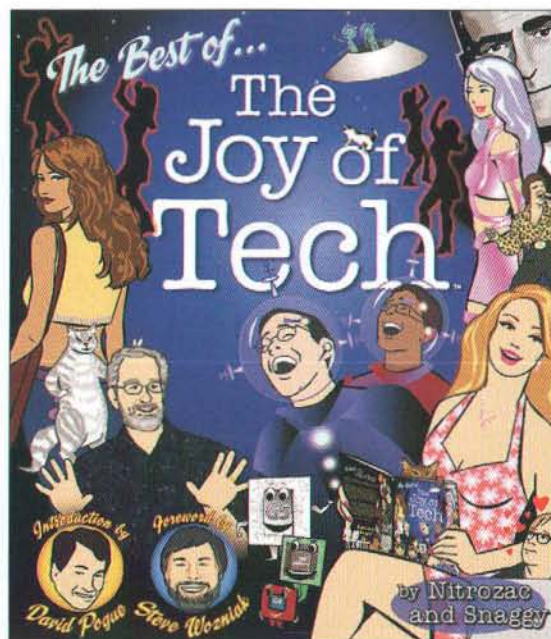
Whither G5?

The G5 desktop system has been around for a few months now, so where do software developers stand on updating their applications for the 64-bit architecture? I decided to get a sample answer by talking to folks whose programs really beat the snot out of hardware. Alias, the makers of Maya, and Discreet, makers

of Combustion (who just released version 3 at the expo). Both companies said that they saw tremendous improvements in performance running their current versions on G5 hardware, without any patches to take advantage of the new hardware advancements. Each also ran into a few compatibility issues, none deal breakers, and all fully documented on their web sites. Last, both Alias and Discreet said they currently have no official plans to update their applications to take full advantage of the G5 architecture, citing the already significant speed boost they currently have. It is something they are always looking at however, so don't expect them to let the advantages of the G5 wait for too long.

DRIVE SAVERS

1-800-440-1904. The one phone number every admin has handy, and prays they never have to call. These folks were at the show, showing off some of the more extreme recovery jobs they've received. Their enterprise business is growing. They are more and more able to handle really big recovery jobs. Don't worry, if you ever need them, they will be there.



Sean Whelan is the Senior Director of Quality Assurance for Commercial Print Products at EFI, Inc. He leads a QA team whose engineers test and release software targeted to customers on Mac and Windows platforms as well as testing hosted Web Products.

MAC OS X PANTHER

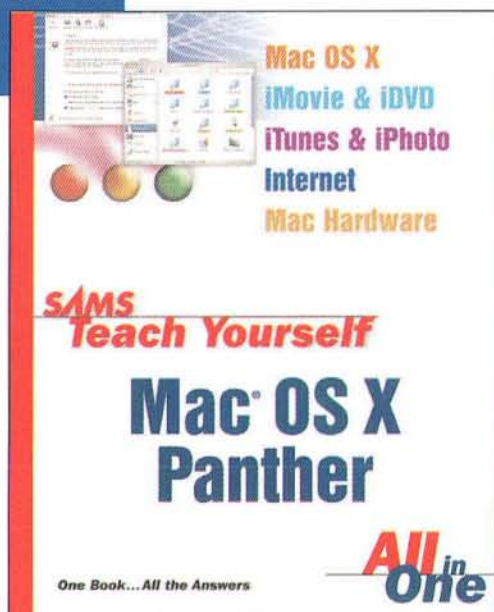
ALL YOU NEED TO KNOW ABOUT PANTHER

Want to know what's new in Panther? Turn to the one book with all the answers.

Sams Teach Yourself Mac OS X Panther All in One

by Robyn Ness and John Ray

ISBN: 0-672-32603-5
\$29.99 US



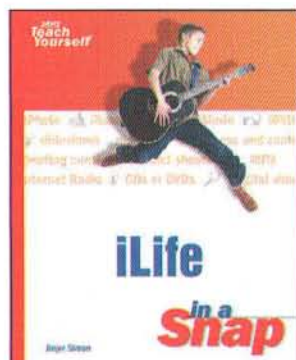
Sams Teach Yourself Mac OS X Panther All in One is designed to teach, in one book, the Mac user how to easily work with the hardware, the operating system, and key applications. Rather than focusing on a single product, the book covers multiple products and technologies together in a logical fashion.

Topics Include:

- Understanding the Mac OS X Panther interface.
- Burning CDs and DVDs with iDVD.
- Playing and organizing MP3s and digital music with iTunes.
- Digital photography with iPhoto.
- Editing digital video with iMovie.

Visit www.amazon.com/samsbooks to download an overview of Panther's new security features

OTHER GREAT SAMS BOOKS ON PANTHER

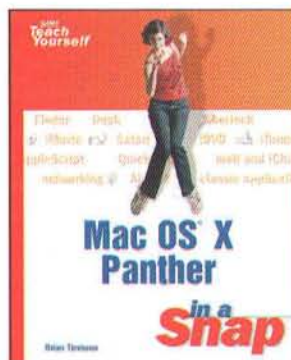


iLife in a Snap

by Jinger Simon

ISBN: 0-672-32577-2 • \$19.99

Available Dec 2003

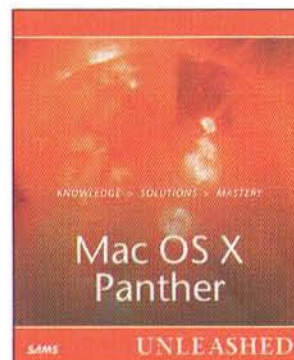


Mac OS X Panther in a Snap

by Brian Tiemann

ISBN: 0-672-32612-4 • \$19.99 US

Available Dec 2003



Mac OS X Panther Unleashed

by John Ray and William Ray

ISBN: 0-672-32604-3 • \$49.99 US

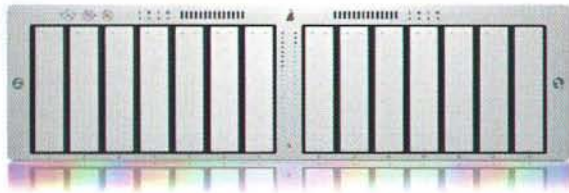
Available Jan 2004

amazon.com

SAMS

www.sampublishing.com

Available at **DEPOT**



DANTZ

The big announcement from Dantz this time around was version 6.0 of Retrospect for Macintosh. A multitude of improvements have been rolled into the new version. High on the list is the removal of the 1 TB limit present in previous version. You can now back up data sets and volumes up to 1,000 TB. Fibre channel support has been added, as well as the ability for backups to span multiple FireWire and USB disks. Obviously, the latest Macs and operating systems are now supported, and Dantz also includes a disaster recovery CD with Mac OS X 10.3.1.

NETOPIA

Timbuktu Pro 7.0

Timbuktu Pro has been the gold standard for remote computer control for years. Version 7.0 only improves on an already great product. Released in December, version 7.0 adds Panther support, including the ability to handle the fast user switching feature of 10.3. You can also now control how much color information is being sent to the controlling computer from the controlling computer. Pop up tip balloons are now part of the program to help you figure out the functions of the not always clear icons. Additionally, some features that in past versions were controlled with AppleScripts are now part of the preferences, making set up and control easier overall.

NetOctopus

Netopia previewed version 5.1 of netOctopus at the show. The main new feature of this will be the Software Delivery System (SDS). It will be added to what Netopia is calling their netOctopus Enterprise Systems Manager. There is a long list of features that this system will over. One new feature that will certainly take a load off your network is the ability to resume interrupted downloads without having the re-download the package from the staging server. The new version will be released by the end of the first quarter.

LACIE

The biggest thing coming from LaCie this show was what they very modestly call the Bigger Disk. It should be available by the time you need this. What it is is a box stuffed with four drives, and the necessary hardware to make it look like a single one terabyte volume. The internal hardware uses RAID 0 and spanning to present the host computer with one logical drive. Hence, the host machine requires no drivers to make use of the disk. It's got three interfaces, FireWire 800, FireWire 400, and

USB 1.1/2.0. It is also equipped with a temperature sensing fan to help keep it cool, as well as running quieter. It'll have a price of \$1199 when available.

PROSOFT

Prosoft was showing off the updates to two of their products, Data Rescue, and Data Backup. Data Backup is a completely new program, built from the ground up by Prosoft. Previously, they had licensed Tri-Backup from Tri-Edre Software. When that agreement ended, they engineered their own version. The other application updated was Data Rescue. This update added the capability to do content based recovery, as well as recover images from digital cameras. These two utilities, along with Data Recycler, are now also bundled into the Data Safety Suite.

NOW SOFTWARE

The folks from Now Software showed their latest update to Now Up-To-Date and Contact, bringing the version to 4.5.1. They added Panther compatibility, as well as stability enhancements and other refinements. They also talked a bit about the next major revision, version 5.0, due later this year. There's nothing official, yet, but it's going to be the biggest, most complete update to the program ever. Keep an eye out for it.

JUST PLAIN COOL

MacWorld always has a lot of consumer oriented products on display. These are some of the absolutely coolest gadgets I came across at the show.

DLO

Back in November, I reviewed Digital Lifestyle Outfitters Transpod all-in-one FM transmitter and car charger. I loved it. I also had two complaints. One, it didn't swivel left and right.





Two, they hadn't yet come out with one for the 3G iPods. Well, I can complain no more. They had on display, and available for sale, the Transpod FM. Not only is it designed for the latest iPods, it has the ability, on the extender arm, to swivel left to right. Add to that a digital tuner, and auxiliary output, and you have the best of in car iPod accessories. They also have a wide variety of other accessories, including some very interesting cases. Check them out at www.everythingipod.com.

Harmony

These guys are great. They had on display their latest offering, the SST-659 Family remote. Designed to be easier for everyone in the house to use, with separate direction and activity keys, making it easier for the younger ones in the family to get things going. It also has a great price point. The one thing you sacrifice on this remote is the included TV listings in the remote. You get the service for two months, then must pay a yearly fee. Still, this device is by far, one of the nicest remotes you can get to feed your need to channel surf. www.harmonyremote.com.

MacAlly

As always, MacAlly had a wide variety of offerings on display. New to their clan of products were a line of backpacks and laptop cases, all quite nice looking. The one thing that caught my eye, however, was their retractable cable kit with multi-function tips. I wasn't able to find out too much about it, and it's not yet even on their website. I am very interested to see something like this that can reduce the cable clutter in my bag, and will be following up on this one.

Altec Lansing

The inMotion portable iPod speaker set was far and away the coolest offering the folks from Altec Lansing had on display at MacWorld. Compact, and well designed, these speakers were specifically designed to support the iPod, and give you room filling sound from a very small form. These speakers run off either A/C power or from four AA batteries. When plugged in, the inMotion will also charge your iPod, while the batteries can run for 24 hours straight. These are an absolute must of the traveling iPod user. www.alteclansing.com

O'Reilly

That name should be known far and wide in the tech community. O'Reilly has, almost literally, a ton of well received tech focused books in publication. At the show, they had them all there, as well as one new title that doesn't quite fall into the techie category (perhaps). The Best of The Joy of Tech is a presentation of the online comic series by Nitrozac and Snaggy. Anyone who has been involved in the Mac community for any amount of time will get a big laugh out of these panels. They also had an excerpt from David Pogue's new book, iLife '04, The Missing Manual, on the floor practically moments after the suite was announced during the keynote address. First impression is that this new addition to the Missing Manual series will uphold the good reputation these books have.

"THE SHOW FLOOR IS CLOSED."

Those are some of the sweetest words you can hear on Friday after a long week of Expo. This year's edition had a lot going for it. There were no really big, gee whiz announcements, but there was a lot going on. What we covered above was only a fraction of all that was written and talked about that week. As for the show itself, the floor, as well as the conference sessions, were well attended. Most everyone, exhibitor, and attendee alike, seemed to feel they got their money's worth out of San Francisco this year.

HelpLogic

The Help Authoring Solution for Mac Developers

Project: Stimulus Help

Add File
Edit Page
Add Topic
Edit Topic
Preview
Publish

Project Files

- fileformats.html
- images
- introduction.html
- license.html
- requirements.html

Help Table of Contents

- Introduction
- System Requirements
- How to Use Stimulus
- Supported Formats
- Audio
 - Bass and Treble
 - Audio Equalizer

Workshop

- Notes
- To Do Items
- Testing
- Feature Requests
- Bug Reports
- Code Snippets
- URL Bookmarks

Easily create help systems for your software applications & web sites from a single source.

Save time with the integrated Workshop, TOC Builder, HTML Editor & Page Templates to quickly generate Apple Help, Web-based Help, UniHelp, PDF & more.

SNEAK PREVIEW

www.ebutterfly.com

electric butterfly

By Dave Wooldridge

Sweetening the Deal

Increasing Software Sales with Purchase Incentives

Last month, we explored ways to enhance your software demos and trialware in the hopes of turning more users into customers. While those various tactics should help increase sales, they certainly won't entice *everyone*. Not every person who downloads your software is going to want or need your product, but what about those holdouts who continue to use unregistered versions? After several sessions, they obviously have found your product to be useful or entertaining, or else it would have ended up in their trash bin long ago. These are the users who are definitely interested in your software, but have been reluctant to take the final purchase step. There are typically three reasons that might prevent a satisfied user from becoming a registered customer:

1. **Price.**
2. **Piracy.**
3. **Procrastination.**

Ah, yes, the three P's preventing purchases (try repeating that tongue-twister multiple times). Developers may recognize these three subjects as contributing factors in the ongoing erosion of software sales. Since the three P's will probably always be part of the software landscape, programmers often turn a blind eye, thinking there is little they can do to battle these global problems. Sure, there will always be users who complain about your prices. And there's no way to prevent piracy (if hackers want to pirate or crack your software, they *will* find a way to do it). As for procrastination, don't we all try to avoid spending money unless absolutely necessary? The key is to change our perception of these problems. Instead of giving up the battle, try to find creative avenues to gain some ground. You may not win over all of those users, but you're sure to capture more sales than if you did nothing.

This month, we'll investigate methods to "sweeten the deal" for those satisfied – yet unregistered – users, with the goal of

finding just the right incentive that finally motivates them to purchase a software license.

PRICING STRATEGIES

The price of your software is a good place to start. If your product has been well-received with good reviews and testimonials, but sales are still below reasonable expectations, you may question the price as a possible deterrent. The natural assumption is that the price may be too high, but you should also take into consideration that the price might be too low.

Too low? Is that even possible? Yes, price directly affects consumer perception of your software. Let's use a pastry shop as an example. A pastry shop decides to place their cinnamon rolls on sale for half price. Since they are normally priced at \$2.00, people flock from all over town to buy them for the special sale price of \$1.00. The shop owners are so pleased with the resulting profits (since each cinnamon roll only costs \$0.10 to make) that they decide to drop the price even lower to \$0.25, in the hopes of selling even more volume! To their surprise, sales sharply decline. No one wanted to buy a cinnamon roll for only a quarter. Why? Because the price was so cheap that people believed there had to be something wrong with the cinnamon rolls. When the price dipped below a reasonable level, the general assumption was that the cinnamon rolls were probably stale, "day-old" inventory.

Many shareware developers fall prey to under-estimating the value of their products. If the competition is selling a similar product for \$39.95, don't sell your product for only \$5.00. Placing one copy of your product on a web server as downloadable shareware removes the costly overhead of manufacturing a boxed retail CD-ROM package, so as developers, we often want to pass the savings onto the customer. Unfortunately, this kind of low-ball pricing strategy will not deliver a windfall of sales, but will instead create the perception that your product is not of the same professional quality as your competitor's. In the eyes of the consumer, price is directly related to quality, reinforced by the popular phrase "you get what you pay for." For \$39.95, customers expect a quality product and support from your competitor.

Dave Wooldridge is the founder of Electric Butterfly (www.ebutterfly.com), the web design and software company responsible for HelpLogic, Stimulus, UniHelp, and the popular developer site, RBGarage.com.

For only \$5.00, the element of risk is introduced with your product. You may be seen as a hobbyist with an uncertain future instead of a stable, professional software company. Sure, it's only \$5.00, so the risk is minimal, but that's \$5.00 that could have been put toward the "safe" purchase of the competition's \$39.95 product.

Beyond the fact that low-end pricing can create inaccurate consumer perceptions, it can also negatively affect the future of your business. If you're selling small utilities that only offer a single feature, then pricing these types of products under \$10 may be appropriate, but for larger multi-featured applications, prices that fall below \$10.00 can often prevent your business from growing. Managing customer support, sales inquiries, product marketing, and programming takes time... and time equals money. If you're currently a one or two person shareware operation, you'll have to sell a dauntingly high number of \$5.00 licenses to generate enough revenue to afford advertising or additional employees.

Doing the math presents a sobering view of your business projections. Say you want to purchase an online banner advertisement on a popular software site. The banner ad campaign (to advertise your \$5.00 product) costs \$600.00 per month. If you utilize an e-commerce partner (such as Kagi, cSellerte, RegNow, RegSoft, DigiBuy, Reg.Net, etc.) who subtracts a processing fee from every software sale (usually an average minimum of \$1.00–\$3.00 per transaction), then for the sake of this example, let's say \$1.00 is the order processing fee. That means for every \$5.00 sale, your business receives a net profit of \$4.00. You would need to sell 150 software licenses in only one month just to break even from the \$600.00 advertising expenditure! Raising your product's price to \$15 or higher drastically reduces the number of licenses you need to sell to pay for the advertising. Even with a \$2.00 order processing fee, a net profit of \$13 requires you to only sell 47 licenses to break even. A \$23.00 net profit (\$25.00 price) requires only 26 licenses to be sold.

I'm not recommending price gauging, by any means. The examples are only to illustrate how your pricing strategy can greatly affect a multitude of business factors, so any decisions to either raise or lower your software prices should not be done impulsively. Don't lower your price because one angry user on an Internet newsgroup condemned your product for not being free. No matter how affordable you make the price, there will always be a select few individuals who complain that it is too high. And don't lower your price in the hopes of curbing piracy. Price usually has nothing to do with piracy. It doesn't matter if your product is \$5.00 or \$500.00, hackers crack software for the love of the challenge. Take the time to study the current marketplace and your competition. The trick is to set a price that effectively conveys quality while still beating competitive prices. If your competitor's product is \$39.95, try pricing your product at a conservative \$24.95. If you offer both a CD-ROM version and

a downloadable version, build the manufacturing cost (let's say \$5.00) into the CD-ROM price while giving a discount to your download customers. In that situation, you may want to try \$29.95 for the CD-ROM version and \$24.95 for the download version. You can always increase the price as you release new versions with additional features.

If you do decide to lower an already established price, you'll need to be careful how you position the change so as not to upset existing customers. If a customer paid the original \$34.95 price last month and discovers that this month, the price has been reduced to \$24.95 for no good reason, then expect to receive some angry e-mails. The saving grace is that customers perceive items going on sale for a limited time as an acceptable reason for lowered prices. Purchasing the item at regular price, not knowing that it would go on sale a month later, will undoubtedly cause irritation, but consumers will not harbor a grudge toward the software developer/publisher.

Declaring a sale price attracts new customers, especially satisfied users who regarded the original price as too high to warrant a purchase. But don't just state the new sale price. Let consumers know exactly what they are saving. **NOW ON SALE:** Only \$24.95 – Save \$10 of the regular price of \$34.95 for a limited time! If those satisfied users have been holding out due to price, remove that barrier with a special sale opportunity that they won't want to miss.

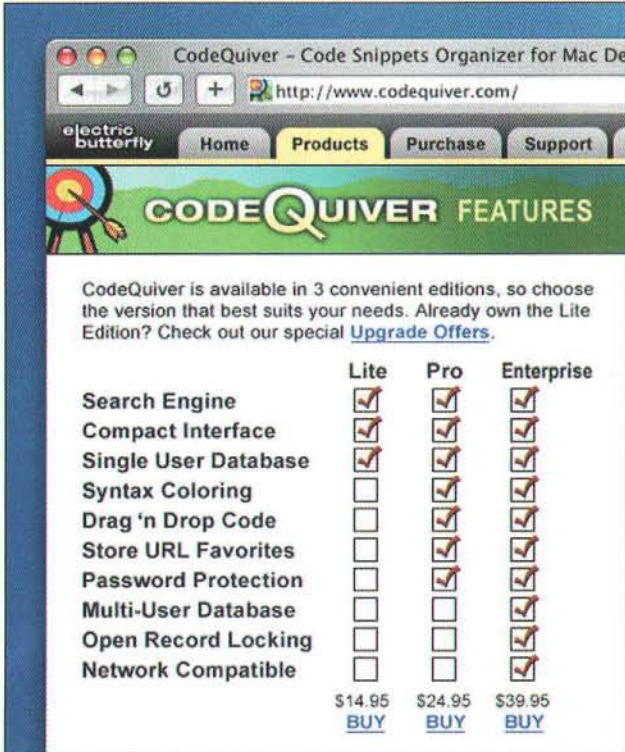
TIERED EDITIONS

On the opposite end of the spectrum, there are software products that sell for several hundred to several thousand dollars each. These applications are usually professional tools geared toward software developers, graphic artists, web designers, musicians, engineers, etc. These products used to be considered niche markets with smaller customer bases, so their expensive price tags would help finance the high cost of developing and supporting such sophisticated tools when the volume of sales couldn't. In today's world of affordable CD/DVD recorders, scanners, digital cameras and camcorders, consumers are finding an everyday use for these high-end software tools. They don't need all of the professional features, so they can't justify paying the high prices. Instead of ignoring this viable audience, many companies are releasing tiered editions of their software to cater to the consumer, prosumer, and professional markets.

Adobe Photoshop has been a longtime favorite among graphics professionals, but its price leaves it far outside the reach of most consumers' budgets. To take advantage of the growing popularity in digital cameras, Adobe released the affordable Photoshop Elements, a striped down version of Photoshop that includes the essential features needed for modifying and enhancing digital photos and scans while simplifying the process with step-by-step wizards and user-friendly guides. Adobe can't lose with this strategy, since Photoshop Elements, either bundled with a scanner/digital

camera or purchased through retail, introduces new users to the power of Photoshop. Consumers will continue to upgrade to the latest version of Photoshop Elements and tech-savvy prosumers will eventually feel ready to graduate to the full-version of Photoshop. Either way, Adobe has created a solution for all levels of expertise, so as not to miss the sales opportunities emerging in the growing digital lifestyle arena.

For business and development tools, it's common to see three tiers: a standard edition, a professional edition, and an enterprise edition. Different companies use different names, but the end result is that they have the ability to offer several versions, each with a distinct price point and feature set, targeted to a specific market (see **Figure 1**). The standard edition may offer only basic features, while the professional edition adds advanced features. The enterprise edition then integrates multi-user collaboration and networking with the pro features.



The screenshot shows a web browser window with the URL <http://www.codequiver.com/>. The page has a navigation bar with links: Home, Products, Purchase, and Support. Below the navigation bar is a header with the text "CODEQUIVER FEATURES" and a logo. The main content area contains a table comparing three editions: Lite, Pro, and Enterprise. The table lists features and their availability across the editions, with prices and "BUY" buttons at the bottom.

	Lite	Pro	Enterprise
Search Engine	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Compact Interface	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Single User Database	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Syntax Coloring	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Drag 'n Drop Code	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Store URL Favorites	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Password Protection	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Multi-User Database	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Open Record Locking	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Network Compatible	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	\$14.95 BUY	\$24.95 BUY	\$39.95 BUY

Figure 1. Depending on your product's feature set, offering tiered editions will attract a broader audience, allowing customers to purchase the version that best suits their needs and budget.

Why go to all the trouble of releasing multiple editions of the same product? Because a one-person home business operator will not want to spend a lot of money on enterprise features that he/she won't use. Don't lose a potential sale. Offer a single-user personal edition to cater to those small business owners. As their business grows, provide an upgrade path, so

that they can easily move up to the next tier if those features become necessary to their business. Giving consumers a choice allows them to purchase the features that best suit their needs and budget.

Even if your software sells for under \$100.00, establishing tiered editions can increase sales. Those satisfied users who refused to pay for unneeded features can finally purchase the appropriate version that meets their comfort level. In **Figure 1**, all three editions and their respective feature sets for our fictional CodeQuiver product are listed on the company web site for users to compare. These kinds of checklists are a great way to easily showcase the differences between the tiered editions. Not including them in your documentation, marketing materials and web site will only invite countless users to e-mail you asking how they are different (which will quickly consume too much of your precious development time).

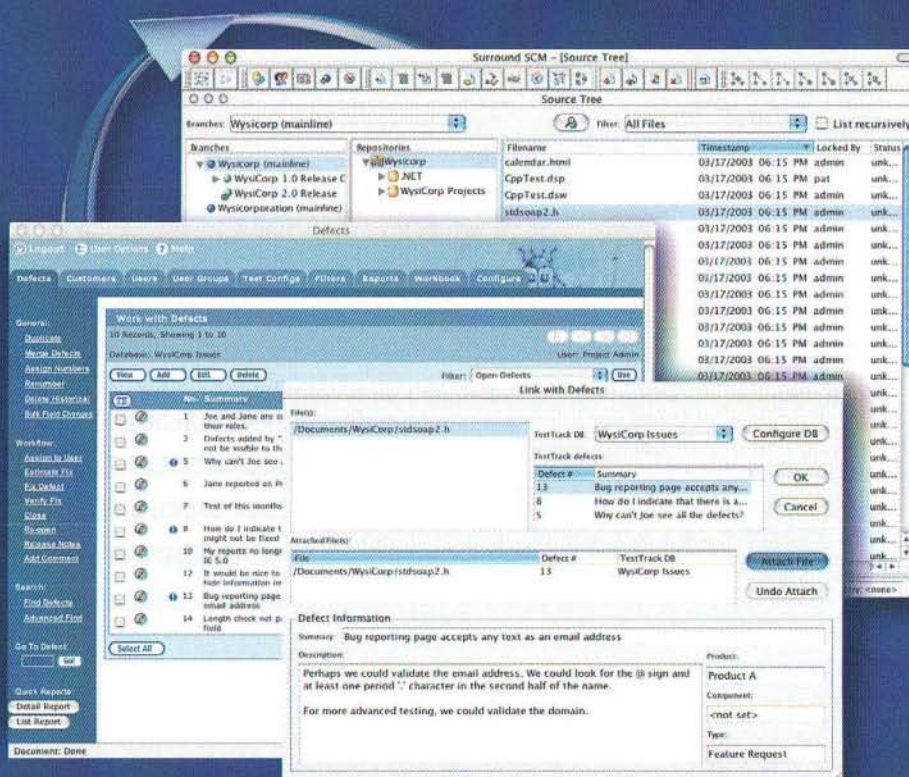
For products whose feature sets can't easily be divided into three marketable editions, another approach is to keep your full product intact and simply offer a stripped-down, free Lite version as a way to attract new users. What makes a free Lite version different than the usual trialware is that being free, word of mouth spreads faster and people are usually more eager to download free tools than those that expire after 30 days. An effective, free Lite edition is one that offers enough features to be useful, but leaves out the most desirable features found only in the full commercial version. Free Lite versions do not display the typical nag screens found in shareware, so their prime objective is to produce loyal users who are constantly recommending your product, driving traffic to your web site. The retail industry calls this a "loss leader" with the goal that people come to download the free Lite version, but while they are visiting your site, end up purchasing the full version or some of your other products. For this strategy to really work, your web site needs to include finely tuned marketing messages that heavily promote the benefits of upgrading to the full commercial version. Offer special upgrade pricing for all Lite users. Why would you give Lite users a discounted upgrade price when they never bought anything to begin with? Simple. Reward customers for being loyal users, since they could just as easily buy your competition's product instead. The sole reason that the Lite version exists is to help sell more software, so make that purchase decision easy for users to make.

If you do decide to offer a free Lite edition, just remember that users will expect you to maintain and update the Lite version. This can be a draining overhead of development and support costs if the Lite version does not effectively increase sales for the full commercial version. On the flip side, it also opens an amazing bundling opportunity for you as well. If your software is an audio/video editor, music jukebox or multimedia viewer that could be used in conjunction with a MIDI keyboard, MP3 player, scanner, digital camera or camcorder, hardware vendors may be interested in bundling

Complete Source Control and Defect Management

 **Seapine Software™**
changing the world
of software development

for Mac OS X



Effective source code control and defect tracking require powerful, flexible, and easy-to-use tools—Surround SCM and TestTrack Pro

- Complete source code control with private workspaces, automatic merging, role-based security, and more
- Comprehensive defect management — track bug reports and change requests, define workflow, customize fields
- Fast and secure remote access to your source files and defects — work from anywhere
- Advanced branching simplifies managing multiple versions of your products
- Link code changes with defects and change requests — know who changed what, when, and why
- Scalable and reliable cross-platform, client/server solutions support Mac OS X, Windows, Linux, and Solaris
- Exchange data using XML and ODBC, extend and automate with SOAP support
- Licenses priced to fit your budget

Seapine Software Product Lifecycle Management
Award winning, easy-to-use software development tools

Seapine
Surround SCM
Seapine
TestTrack PRO



**Download Surround SCM
and TestTrack Pro at**
www.seapine.com
or call 1-888-683-6456

all product names listed herein are registered trademarks of their respective owners. All rights reserved.



your software with their products. The catch is that most manufacturers will not bundle time-limited trialware or crippled demos – they want working software that won't expire, so they can advertise that their hardware products include full-functioning software that enhances the user experience. Getting your free Lite version distributed as bundled software is a great way to introduce new users to your product. Once they are registered users of the Lite version, you can offer them special upgrade opportunities (for the full version) to get even more out of their software experience!

DANGLING THE CARROT

With cracked software and stolen serial numbers so easy to find on the Internet, simply offering a sale price may not motivate users of pirated software to actually buy your product. As stated earlier, no matter how low you price your product, if hackers want to crack your application, they'll do it for the love of the challenge, not to save money. Don't waste your time trying to find ways to convert hackers into customers. Your efforts should be directed at the casual consumers who stumble upon pirated software and serial numbers on the various warz sites. Those who feel guilty might decide to take advantage of a sale price in order to replace their cracked version with a legitimate copy, but with the popularity of file sharing, don't hold your breath waiting. Sure, you could spend lots of time blacklisting stolen serial numbers on warz sites and sending warning e-mails to illegal users. While those defensive measures may stem the tide of piracy just a little, they won't sell more licenses.

The screenshot shows a web browser window titled "CodeQuiver - Code Snippets Organizer for Mac Developer". The address bar shows "http://www.codequiver.com/". The website has a navigation bar with links: Home, Products, Purchase, Support, and New. The main content area features a "CODEQUIVER" logo and a "Special Limited-Time Offer!" section. The offer text reads: "Upgrade or Purchase CodeQuiver 1.5 before March 1, 2004 & receive the CQ Solutions Library FREE! (a \$50 value)". A yellow badge with a red border says "ONLY 9 DAYS LEFT". Below the offer text, there are two buttons: "Buy Now" (Only \$24.95 USD) and "Upgrade" (Only \$14.95 USD). A "More Info >>" link is also present. The text below the buttons describes the CQ Solutions Library, stating it includes more than 500 code snippets for various programming languages and provides test-driven, ready-to-use functions, routines, declares, and scripts in a convenient drag 'n drop library format.

Figure 2. While the free extras add value to the purchase price, this fictional offer also includes an expiration date as an incentive to place an order sooner rather than later.

So how do you convince a satisfied user to purchase your software instead of downloading a pirated version? Desperate times call for creative solutions. Offer something to consumers that cannot be found on a warz site or file sharing service. In **Figure 2**, as a value-added bonus for upgrading or purchasing our fictional CodeQuiver product, customers receive a Solutions Library of more than 500 code snippets in various programming languages. What's nice about this particular bonus is that the items come formatted as a database library that natively integrates with CodeQuiver, strengthening the usefulness of the product (which is a code snippets organizer). Plus, as a digital file, it can be easily delivered as a download, so no extra shipping costs are required. Other downloadable items that work well as value-added extras are ebooks, plugins, related helper utilities, etc. To effectively attract customers, make sure the bonus item is related to your product in some way, so that it also serves to enhance their experience with your software. For example, a music production tool could include a free drum loops collection or audio effects plugins with every purchase.

To help prevent digital bonus items (such as CodeQuiver's Solutions Library) from ending up as a pirated commodity as well, do not include it with the licensed download of the product. Require eligible customers to register their license online to gain access to a special password-protected download site. Encrypt the download with the customer's serial number or some similar method that would convince customers that illegally distributing this bonus item could be traced back to them. Obviously, an experienced hacker would see through the protective guise, but the purpose of this subtle scare tactic is to prevent average customers from making the bonus item available on file sharing services, etc.

If your product is only available as an electronic download, then you may want to avoid offering bonus items that require physical shipping, such as printed books, training videos or apparel. Unless you are already shipping a CD-ROM to your customers, the added expense of packaging and mailing the bonus item will drastically reduce your profit margin (especially if you are shipping it overseas). You may not make enough money selling a \$39.95 license to a customer in Italy if it costs \$26.18 to ship the free bonus book across the Atlantic Ocean.

Offering free software that you've created or some value-added item that does not incur manufacturing costs or licensing/royalty fees will also help you preserve your profit margin. For example, **Figure 2**'s Solutions Library was compiled by CodeQuiver's creators, so bundling it with CodeQuiver purchases only cost the developers the initial time it took to produce the bonus library. Since CodeQuiver is a relatively inexpensive product, a Solutions Library made by a third-party developer that requires a licensing fee or royalty payment for each unit sold can negatively impact the net profit

of each sale. Calculate the cost of goods before finalizing a special offer to ensure that your profit margin remains acceptable. The goal is to increase sales so that you can grow your business, not go out of business.

Do not offer product documentation or support as the free bonus items. These are elements that customers expect to automatically receive with a paid license. The only time support qualifies as a value-added item is if you offer paid support plans as separate purchases. For example, many development tools include a certain level of basic support, but beyond the initial offering, paid premium support plans are available for customers who need ongoing priority assistance. If your marketing efforts are targeting professional businesses, offering your Gold Support Plan (a \$300.00 value) free with every purchase of your product might be quite attractive to business executives, engineers, programmers, etc.

Always state the monetary value (even if estimated) of your bonus items. Informing consumers of exactly how much money they would be saving by taking advantage of your amazing offer can be the key factor that convinces them to buy. A user may not know that the free audio effects plugins that come bundled with every purchase of your music production software are worth a combined retail value of \$900.00. They knew they were saving money, but they may not have been aware that they would be saving *that much* money! With this in mind, make sure the bonus item you're offering with every purchase is an appropriate sales incentive. Including a free t-shirt (a \$12.00 value) with the purchase of your \$850.00 animation software may not provide enough motivation for those undecided users.

Another popular selling technique is the bundling of several related products into one cost-effective suite or collection. This approach tends to work well for productivity tools and games. Macromedia bundles Dreamweaver, Flash, Fireworks and Freehand together in one Studio package. Adobe's Creative Suite includes Photoshop, Illustrator, InDesign, and other applications. The hook is that the collection is offered at a price that's much less than if you were to buy all of those applications individually. To consumers who are interested in one or more of those products, the bundled suite appears to be best deal. At first glance, some developers may disregard this strategy, thinking that the suite cannibalizes individual product sales, but they would be wrong. Don't assume that someone who purchased the Suite would have ordinarily bought each and every product at full price if the Suite did not exist. Interested consumers would have probably purchased only one or two products, so they end up spending less than the Suite price. But since the Suite is such a good deal, they are willing to spend more money to save money. This introduces new users to products they ordinarily would not have purchased. If they enjoy using the software, they become loyal users who end up purchasing the next Suite upgrade instead of only upgrading the one primary product they use most. In the long run, it's a win-win situation for everyone: customers save money and you net higher profits.

FOR A LIMITED TIME ONLY!

Consumers who are interested in your software, but keep putting off the purchase for another day will continue to stall as long as there is no rush. Unless they urgently need your software for a specific task, they will not be in a hurry to spend their hard-earned money if your special offer or bundled suite will always be available. Procrastination leads to forgetfulness, so give them a reason to purchase before their interest wanes. By placing an expiration date on your special offer, you're forcing those users to make a decision within a specific time window. The fear of missing a money-saving opportunity is a powerful purchase incentive. In **Figure 2**, our fictional CodeQuiver web site takes it a step further by counting down the number of days left to take advantage of the special offer. Experienced web site designers can use scripting languages like PHP or JavaScript to create a dynamic countdown, so that with each new day, the displayed countdown on your site automatically calculates the number of days left before the special offer ends.

If you maintain an opt-in e-mail newsletter for announcing software news to your customers and interested users, be sure to utilize this powerful marketing tool to remind subscribers that your special offer will end soon. Do not assume that people will remember on their own. We all lead extremely busy lives, so there's nothing wrong with a helpful reminder of a money-saving opportunity. Just don't beat them over the head with a daily countdown or you may see a record number of people unsubscribing from your e-newsletter (or worse: accusing you of e-mail marketing abuse). Save the countdown for your web site. If your special offer lasts two months, then send your subscribers an e-mail reminder once a month and then one last reminder a week before the offer ends. Only send e-mails to those people who voluntarily subscribed to your e-newsletter. With the new U.S. spam law now in effect, developers need to tread carefully when promoting software via e-mail. We'll discuss the guidelines of the new CAN-SPAM Act and how it affects software developers in a future column.

IDENTIFYING VALUE

We've covered several strategies for "sweetening the deal," but you know your software products and users better than anyone. Look at customer feedback and feature requests for inspiration. Devise your own purchase incentives that work for your software products. What value do your users place on price, free bonus software, choice of features? Experiment to find that key factor or combination of elements that persuades them to buy. If you can motivate those holdout users to finally purchase a license (and still make a decent profit in the process), then you'll be one step ahead of the game with the increased revenue needed to grow your software business.

by Tim Monroe

Swing Shift

Editing QuickTime Movies with Java

INTRODUCTION

In the previous *QuickTime Toolkit* article (“Krakatoa, East of Java” in *MacTech*, January 2004), we saw how to use QuickTime for Java to open and display QuickTime movies in a Java-based application. We saw how to use AWT components to elicit movie files from the user (albeit indirectly, by a call to the `QTFile` method `standardGetFilePreview`) and to display and manage movies in a window (using the `Frame` and `QTComponent` components). We also saw how to display and handle the application’s menus and menu items (using the `Menu`, `MenuItem`, and `Action` classes).

In this article, we’re going to extend the application we built in the previous article — called *JaVeez* — to handle movie editing. We’ll see how to do standard cut, copy, and paste editing, and we’ll see how to save an edited movie into its original file or into a new file. We also need to implement the standard document behaviors, such as prompting the user to save or discard any changes when an edited movie window is about to be closed. In particular, we’ll see how to display and manage the “Save Changes” dialog box shown in **Figure 1**.



Figure 1: The Save Changes dialog box of *JaVeez*

By sheer coincidence, the dialog boxes we’ll use to perform these remaining tasks are best displayed using Swing, not AWT.

The dialog box in **Figure 1** can be displayed with a single line of code. We get the warning icon essentially for free, by passing the appropriate constant. We also get with the standard behavior where the Save button is highlighted as the default button and is activated by hitting the Return or Enter key on the keyboard. To my knowledge, it’s not possible to designate a default button when constructing a dialog box using AWT.

We’ll also take a look at handling several of the items in the Mac OS X Application menu. We’ll see how to display an About box when the user chooses the “About *JaVeez*” menu item, and we’ll see how to handle the Quit menu item correctly. We’ll use a class introduced in J2SE 1.4.1 for Mac OS X, `Application`, to handle these operations. This class also makes it easy for us to handle basic Apple events, such as the OpenDocuments event that is sent to our application when the user drops some movie files onto the application icon in the Finder. By the end of this article, *JaVeez* will be virtually indistinguishable from the C-language Carbon-based application *QTShell* it’s modeled upon.

MOVIE EDITING

It’s extremely easy to add support for movie editing to our application, because the `MovieController` class in QuickTime for Java provides pretty much all the methods we need. For instance, to cut the current movie selection, we can execute the `cut` method on the movie controller object associated with a movie window. As we’ll see in a moment, we also need to place the cut segment onto the system scrap so that it’s available for subsequent pasting (either by *JaVeez* itself or by any other QuickTime-savvy application).

In order for these editing methods to work properly, we need to explicitly enable editing. You may recall that in the previous article, we executed this code in the `createNewMovieFromFile` method:

```
if ((mc.getControllerInfo() &
    StdQTConstants.mcInfoMovieIsInteractive) == 0)
    mc.enableEditing(true);
```

This code looks to see whether the movie is an interactive movie (such as a QuickTime VR movie or a Flash movie); if it’s interactive, it is not editable and therefore we do not want to

Tim Monroe is a member of the QuickTime engineering team at Apple. You can contact him at monroe@mactech.com. The views expressed here are not necessarily shared by his employer.

enable editing. (If we did try to enable editing on an interactive movie, an exception would be raised.)

Handling Editing Operations

In JaVeez, the user performs editing operations using either the menu items in the Edit menu or their standard keyboard shortcuts. In either case, one of our action listeners will be invoked. For instance, **Listing 1** shows our declaration of the `UndoActionClass`, an instance of which is attached as a listener to the Undo menu item and its keyboard equivalent. As you can see, we simply call the movie controller's `undo` method and then call our own method `updateEditedWindow`.

Listing 1: Undoing the previous edit

```

UndoActionClass
public class UndoActionClass extends AbstractAction {
    public UndoActionClass (String text, KeyStroke shortcut) {
        super(text);
        putValue(ACCELERATOR_KEY, shortcut);
    }
    public void actionPerformed (ActionEvent e) {
        try {
            mc.undo();
            updateEditedWindow();
        } catch (QTEException err) {
            err.printStackTrace();
        }
    }
}

```

The `updateEditedWindow` method performs any operations that might be required when that the movie in the specified frame has been altered by an edit operation. For instance, the size of the movie may have changed, so we'll want to call our method `sizeWindowToMovie` to ensure that the movie window is sized to exactly contain the movie and the movie controller bar (if it's visible). Also, we'll want to adjust the items in the File and Edit menus appropriately. **Listing 2** shows the `updateEditedWindow` method.

Listing 2: Updating the state of an edited window

```

updateEditedWindow
public void updateEditedWindow () {
    try {
        mc.movieChanged();
        mc.movieEdited();
        mc.controllerSizeChanged();
        adjustMenuItems();
        sizeWindowToMovie();
    } catch (QTEException err) {
        err.printStackTrace();
    }
}

```

The actions associated with pasting and clearing can be handled by code that's exactly analogous to that in **Listing 1** (just change "undo" to "paste" or "clear" in the `actionPerformed` method). In the two remaining cases, cutting and copying, we also need to make sure to move the cut or copied segment to the scrap, by calling the `putOnScrap` method. **Listing 3** gives our implementation of the cut operation. The movie controller's cut method returns a QuickTime movie that holds the segment cut from the movie in the window; we copy it to the scrap and then dispose of that movie segment by calling `disposeQTObject`.

Listing 3: Cutting the movie selection

```

CutActionClass
public class CutActionClass extends AbstractAction {
    public CutActionClass (String text, KeyStroke shortcut)
    {
        super(text);
        putValue(ACCELERATOR_KEY, shortcut);
    }
    public void actionPerformed (ActionEvent e) {
        try {
            Movie editMovie = mc.cut();
            editMovie.putOnScrap(0);
            editMovie.disposeQTObject();
            updateEditedWindow();
        } catch (QTEException err) {
            err.printStackTrace();
        }
    }
}

```

Our action handler for the copy operation is even simpler, since we don't need to call `updateEditedWindow` (because copying the current selection from a movie does not change the original movie).

Selecting All or None of a Movie

As usual, our Edit menu contains two further items, "Select All" and "Select None", which are quite easy to implement. In earlier articles, we've seen how to handle these items by calling `MCDoAction` with the `mcActionSetSelectionDuration` selector. **Listing 4** shows how JaVeez handles the "Select All" command.

Listing 4: Selecting all of a movie

```

SelectAllActionClass
public class SelectAllActionClass extends AbstractAction {
    public SelectAllActionClass (String text,
                                KeyStroke shortcut) {
        super(text);
        putValue(ACCELERATOR_KEY, shortcut);
    }
    public void actionPerformed (ActionEvent e) {
        try {
            TimeRecord tr = new TimeRecord(m.getTimeScale(), 0);
            mc.setSelectionBegin(tr);
            tr.setValue(m.getDuration());
            mc.setSelectionDuration(tr);
        } catch (QTEException err) {
            err.printStackTrace();
        }
    }
}

```

The interesting thing here is that a `TimeRecord` is an object (which must be explicitly constructed by a call to `new`), not a structure as in our C code. In general, Java does not support either `struct` or `union`; instead, we need to build these sorts of composite types using classes or interfaces. The package `quicktime.std.clocks` provides access to `TimeRecord` objects and the methods to get and set their properties.

Listing 5 shows how JaVeez handles the "Select None" command.

Listing 5: Selecting none of a movie

```

SelectNoneActionClass
public class SelectNoneActionClass extends AbstractAction {
    public SelectNoneActionClass (String text,

```



```

KeyStroke shortcut) {
    super(text);
    putValue(ACCELERATOR_KEY, shortcut);
}
public void actionPerformed (ActionEvent e) {
    try {
        TimeRecord tr = new TimeRecord(m.getTimeScale(), 0);

        mc.setSelectionDuration(tr);
    } catch (QTErrException err) {
        err.printStackTrace();
    }
}
}

```

Enabling and Disabling Menu Items

Editing a movie usually entails that some of the items in the Edit and File menus need to be adjusted. For instance, cutting a segment out of a movie should be undoable, so we want to make sure that the Undo menu item is enabled. Similarly, once we've made any edit operation whatsoever to a movie, we want to make sure that the Save menu item is enabled. **Listing 6** shows our implementation of the `adjustMenuItems` method. Notice that we disable the entire Edit menu if editing is not enabled for the movie. Also, we can use a movie's `hasChanged` method to determine whether to enable or disable the Save and Undo menu items. (As you would guess, we clear the movie's changed state — by calling the `clearChanged` method — when the user saves a movie.)

Listing 6: Adjusting the menu items

```

public void adjustMenuItems () {
    try {
        if ((mc.getControllerInfo() &
            StdQTCConstants.mcInfoEditingEnabled) == 0)
            editMenu.setEnabled(false);
        else
            editMenu.setEnabled(true);

        if (m.hasChanged()) {
            miSave.setEnabled(true);
            miUndo.setEnabled(true);
        } else {
            miSave.setEnabled(false);
            miUndo.setEnabled(false);
        }

        if (mc.getVisible())
            miShowController.setLabel
                (resBundle.getString("hideControllerItem"));
        else
            miShowController.setLabel
                (resBundle.getString("showControllerItem"));
    } catch (QTErrException err) {
        err.printStackTrace();
    }
}

```

It's worth remarking that this menu-enabling logic is quite a bit simpler than that contained in our C-language applications, primarily because in JaVeez, a menu is always associated with a movie window. This means that the Close and "Save As..." items should always be enabled.

FILE MANIPULATION

Now let's consider how to handle the Close and "Save As..." menu items, along with the Save and Quit menu items. When

handling the Close and Quit items, we need to check to see whether the user has made any changes to the frontmost movie window or to any of the open movie windows. If so, we need to give the user an opportunity to save or discard those changes. We also need to allow the user to cancel the close or quit operation altogether.

Closing a Movie Window

We can do all this by adding a *window adapter* to our application. A window adapter is an object that listens for specific events involving a window and executes a method when it receives a notification that one of those events has occurred. Currently, we can listen for window activation or deactivation, iconification or deiconification, and opening or closing. There are two flavors of window closing events; we can be notified that a window is in the process of being closed, or we can be notified that a window actually has closed. Clearly, we want to listen for the first kind of window closing event so that we can cancel it if necessary.

We'll define a class that extends the `WindowAdapter` class; this means that our class needs to implement some but not necessarily all of the methods in the `WindowAdapter` class. As just mentioned, we want to implement only the `windowClosing` method. **Listing 7** shows our definition of the `WindowAdpt` class.

Listing 7: Handling a request to close a window

```

class WindowAdpt extends java.awt.event.WindowAdapter {
    public void windowClosing (java.awt.event.WindowEvent event) {
        try {
            if (m.hasChanged())
                askSaveChanges(resBundle.getString("closeText"));
            else
                dispose();
        } catch (QTErrException err) {
            err.printStackTrace();
        }
    }
}

```

This is simple enough: if the movie in the window has changed since it was opened or last saved, ask the user to save or discard those changes; otherwise, call `dispose` to close the window. (The `dispose` method is implemented by AWT's `Window` class.)

In the `JaVeez` constructor, we create an instance of this class and set it as the window listener like this:

```

WindowAdpt WAdapter = new WindowAdpt();
addWindowListener(WAdapter);

```

All that remains is to write the `askSaveChanges` method. This method needs to display the Save Changes dialog box (see **Figure 1** again) and respond to the user's selecting one of the three buttons in that box. Happily, Swing provides the `JOptionPane` class that is tailor-made for this purpose. **Listing 8** shows our implementation of `askSaveChanges`. We pass in a string object that indicates whether the user is closing a particular window or quitting the application.



See Dick. See Dick run his business with software that wasn't written and designed for his Macintosh. Poor Dick.

A moment of silence for Dick, please. A good guy with a good small business, but his accounting software was one of those PC transcription jobs, not pure MAC like MYOB AccountEdge and MYOB FirstEdge.

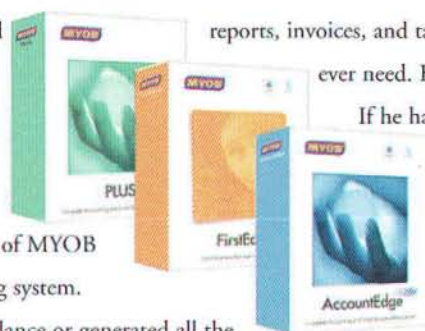
If only he'd known about the amazing capacity of MYOB software to bring out the best in his MAC operating system. He could have tracked and managed finances at a glance or generated all the

reports, invoices, and tax documents that he and his accountant would ever need. He could have spent more time with his clients.

If he had only known that MYOB develops the world's best selling MAC small business management software for lots of good reasons, this story might have had a happy ending. Sorry Dick.

**MYOB,
THE MAC ANSWER.**

©2003 MYOB US, Inc. (800)322-MYOB www.myob.com/us



Listing 8: Prompting the user to save a changed movie

```
askSaveChanges
public boolean askSaveChanges (String actionString) {
    Object[] options = {"Save", "Cancel", "Don\u00A9t Save"};
    boolean windowClosed = false;

    int result = JOptionPane.showOptionDialog(this,
        "Do you want to save the changes you made "
        + "to the document"
        + "\n\u201C" + baseName + "\u201D?",
        "Save changes before " + actionString,
        JOptionPane.YES_NO_CANCEL_OPTION,
        JOptionPane.WARNING_MESSAGE,
        null,
        options,
        options[0]);

    if (result == SAVE_RESULT) {
        // save the changes and close the window
        save();
        windowClosed = true;
    } else if (result == CANCEL_RESULT) {
        // don't save changes and don't close the window
        windowClosed = false;
    } else if (result == DONTSAVE_RESULT) {
        // don't save changes but do close the window
        windowClosed = true;
    }

    if (windowClosed)
        dispose();

    return(windowClosed);
}
```

The `showOptionDialog` method returns the index in the `options` array of the selected button. `JaVeez` defines these constants to make our code easier to read:

```
private static final int SAVE_RESULT      = 0;
private static final int CANCEL_RESULT    = 1;
private static final int DONTSAVE_RESULT  = 2;
```

When it exits, the `askSaveChanges` method returns a `boolean` value that indicates whether it actually closed the window it was asked to close. Our `windowClosing` override method ignores that value, but we'll need to use it when we call `askSaveChanges` during the process of quitting the application. More on that later.

One final point: what's with the gibberish in a couple of the strings in `askSaveChanges`? Strings in Java are Unicode strings, and using the standard keyboard entries for the apostrophe and the left and right double quotation marks would yield a less satisfactory dialog box. Compare **Figure 1** to **Figure 2**:



Figure 2: The Save Changes dialog box of `JaVeez` (bad characters)

The Unicode value `"\u00A9"` gives us the nicer-looking apostrophe in the word "Don't", and the values `"\u201C"` and `"\u201D"` give us the nicer-looking quotation marks around the filename (though it's hard to tell that in the font used in this dialog box).

Saving a Changed Movie

The `askSaveChanges` method (**Listing 8**) calls `JaVeez`' `save` method if the user elects to save the changes to the window. It's easy to implement that method, using the movie's `updateResource` method. **Listing 9** shows our `save` method.

Listing 9: Saving a movie

```
save
public void save () {
    try {
        if (omf == null) {
            saveAs();
        } else {
            m.updateResource(omf,
                StdQTConstants.movieInDataForkResID, null);
            m.clearChanged();
        }
    } catch (QTException err) {
        err.printStackTrace();
    }
}
```

If no movie file is yet associated with the movie window, we call the `saveAs` method instead of `updateResource`. This has the effect of displaying the file-saving dialog box, which allows the user to specify a filename for the movie.

Saving a Movie into a New File

When the user selects the "Save As..." menu item in the File menu (or, as we just saw, elects to save a movie that has not yet been associated with a movie file), we need to elicit a location for the new movie file and then save the movie into that file. In `JaVeez`, we'll execute this line of code:

```
fd = new FileDialog(this, "Save: JaVeez", FileDialog.SAVE);
```

The AWT `FileDialog` class, when passed the `FileDialog.SAVE` parameter, displays a dialog box like the one shown in **Figure 3**.

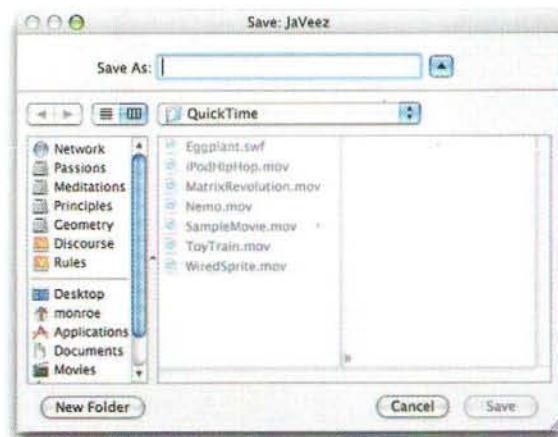


Figure 3: The file-saving dialog box displayed by AWT

Technological.

The logical way to connect your
news and information to the people who count.

At PR Newswire, our approach to distributing technology news is simple: target the people with a real interest in your news – journalists, investors, analysts and consumers – and deliver your message to them in the way they prefer.

PR Newswire has earned the respect of the media and financial community by consistently delivering accurate, credible, reliable news and information directly to their desktops. We also offer a full range of customized solutions to help get your message across with photos, VNRs, Webcasting, and more.

Make sure your news is seen in all the right places.

Call PR Newswire at 888-776-0942 or visit us at www.prnewswire.com.

We tell your story to the world.™



PR Newswire
United Business Media

This is one of the few cases where the AWT dialog box is preferable to the corresponding Swing dialog box, which is shown in **Figure 4**.

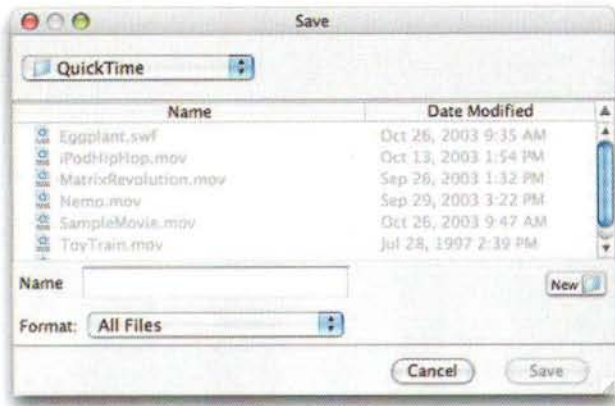


Figure 4: The file-saving dialog box displayed by Swing

As you can see, the AWT box more closely resembles the native file-saving dialog box displayed by Carbon or Cocoa applications running on Mac OS X.

Once the user has selected a new location and file name, we can save the movie into that file by executing the `flatten` method on the movie, as shown in **Listing 10**.

Listing 10: Saving a movie into a new file

```
public void saveAs () {
    try {
        if (fd == null)
            fd = new FileDialog(this, "Save: JaVeez",
                               FileDialog.SAVE);

        fd.setFile(null); // clear out the file name
        fd.show();

        String dirName = fd.getDirectory();
        String fileName = fd.getFile();

        if ((dirName != null) && (fileName != null)) {
            QTFile qtf = new QTFile(dirName + fileName);

            m.flatten(
                StdQTConstants.flattenForceMovieResourceBeforeMovieData,
                qtf,
                StdQTConstants.kMoviePlayer,
                IOConstants.smSystemScript,
                StdQTConstants.createMovieFileDeleteCurFile,
                StdQTConstants.movieInDataForkResID,
                qtf.getName());

            // close the connection to the current movie file
            if (omf != null) {
                omf.close();
                omf = null;
            }

            // now open the new file in the current window
            createNewMovieFromFile(qtf.getPath(), true);
        }
    } catch (QTException err) {
        if (err.errorCode() != Errors.userCanceledErr)
            err.printStackTrace();
    }
}
```

The standard behavior of a "Save As..." operation is that the new movie will replace the existing movie in the existing movie window. Accordingly, `saveAs` closes the connection to the existing movie file (by calling its `close` method) and calls `createNewMovieFromFile` with the full pathname of the new movie file. By passing `true` as the second parameter, we instruct `createNewMovieFromFile` not to reposition the window.

Quitting the Application

When the user decides to quit JaVeez (typically by selecting the Quit item in the Application menu or — on Windows — the Exit item in the File menu), we want to loop through all open movie windows and call the `askSaveChanges` method on each window that's been edited since it was opened or last saved. To my knowledge, AWT does not provide a way to find just the open movie windows. It does provide the `getFrames` method in the `Frame` class, but my experience is that the array returned by `getFrames` includes an entry for all frames ever created by the application (not just the ones that are currently open and visible). Nonetheless, we can use the `getFrames` method to good effect by simply ignoring any frames that are not visible or do not belong to the `JaVeez` class. **Listing 11** shows our implementation of `attemptQuit`.

Listing 11: Handling a request to quit the application

```
public boolean attemptQuit () {
    // try to close all open document windows; if none is kept open, quit
    Frame[] frames = Frame.getFrames();
    boolean didClose = true;
    String quitString = resBundle.getString("quitText");

    // try all open movie windows other than the current one
    for (int i = 0; i < frames.length; i++) {
        if ((frames[i] != this) && (frames[i] != null) &&
            (frames[i].getClass().getName() == "JaVeez") &&
            (frames[i].isVisible())) {
            try {
                JaVeez jvz = (JaVeez)(frames[i]);

                if (jvz == null)
                    continue;

                if (jvz.isDirty())
                    didClose = jvz.askSaveChanges(quitString);

                if (!didClose)
                    return(false);
            } catch (Exception err) {
                err.printStackTrace();
            }
        }
    }

    if (this.isDirty())
        didClose = this.askSaveChanges(quitString);

    if (didClose)
        goAway();

    return(didClose);
}
```

You'll notice that we postpone handling the frontmost window until all other movie windows have been handled. I suspect it's not a good thing to dispose of an object while it's still executing one of its methods.

If none of the open movie windows has been edited or the user does not cancel the closing of any of them, then `attemptQuit` executes the `goAway` method. This method simply closes the application's connection to QuickTime and then exits (**Listing 12**).

Listing 12: Quitting the application

```
public void goAway () {  
    try {  
        if (qtc != null)  
            qtc.setMovieController(null);  
    } catch (QTEException err) {  
    }  
  
    QTSession.close();  
    System.exit(0);  
}
```

APPLICATION OBJECTS

Let's finish up by considering how to integrate our application with the native Mac OS X operating environment so that it acts as much as possible like a well-written Carbon or Cocoa application. Java 1.4.1 for Mac OS X includes the new package `com.apple.eawt`, which contains the `Application` class. This class provides methods that allow us to fine-tune the appearance and behavior of the Application menu, respond to items in that menu, and handle the basic application-related Apple events. By creating an *application object* and attaching to it an *application listener*, we can (for instance) respond appropriately when the user drags some movie files onto our application's icon in the Finder.

To take advantage of these enhancements, we need to import the appropriate packages:

```
import com.apple.eawt.*;
```

Creating an Application Object

You'll recall from the previous article that JaVeez declares the static variable `fApplication`:

```
private static Application fApplication = null;
```

In its constructor method, JaVeez calls the `createApplicationObject` method, which is defined in **Listing 13**. After ensuring that `fApplication` hasn't already been set to a non-null value and that the application is running on Mac OS X, `createApplicationObject` retrieves the application object, disables the Preference menu item, and installs an application listener. The listener implements methods that handle some of the items in the Application menu (About, Preferences, and Quit); these methods are also called when certain Apple events are targeted at the application.

Listing 13: Creating an Application object

```
public void createApplicationObject () {  
    if ((fApplication == null) &&  
        QTSession.isCurrentOS(QTSession.kMacOSX))) {  
        fApplication = Application.getApplication();  
        fApplication.setEnabledPreferencesMenu(false);  
        fApplication.addApplicationListener(new  
            createApplicationObject
```



Fetch

Play Fetch with a panther.

X

Fetchsoftworks.com

(Running dog included.)


```

        com.apple.eawt.ApplicationAdapter() {

    public void handleAbout (ApplicationEvent e) {
        about();
        e.setHandled(true);
    }

    public void handleOpenApplication (ApplicationEvent e) {
    }

    public void handleOpenFile (ApplicationEvent e) {
        launchedFromDrop = true;

        JaVeez jvz = new JaVeez(e.getFilename());
        jvz.createNewMovieFromFile(e.getFilename(), false);
        jvz.toFront();
    }

    public void handlePreferences (ApplicationEvent e) {
        preferences();
        e.setHandled(true);
    }

    public void handlePrintFile (ApplicationEvent e) {
    }

    public void handleQuit (ApplicationEvent e) {
        boolean allWindowsClosed;

        allWindowsClosed = attemptQuit();
        e.setHandled(allWindowsClosed);
    }
}
}

```

These handlers can call the `setHandled` method to indicate whether the event was handled. The most interesting case is the `handleQuit` method, which (as you can see) returns the value it gets from the `attemptQuit` method. This allows our application to prevent itself from quitting if the user cancels the save-or-discard-changes query.

Handling Dragged Files

In order for the user to be able to drop QuickTime movie files onto our application's icon, we need to set the document types supported by JaVeez. We do this by selecting the JaVeez target in the project window and then choosing the "Get Info" item in the Project menu. Click the "+" button at the lower-left corner of the Info window to add the desired file types. As you can see in **Figure 5**, JaVeez supports QuickTime movie files and Flash files.

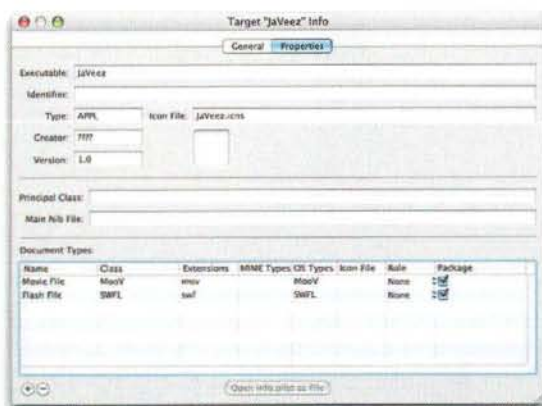


Figure 5: The supported document types

Showing the About Box

We have to tackle one final task, namely displaying our application's About box. We want this to look as much as possible like the About box of QTShell, shown in **Figure 6**.



Figure 6: The About box of QTShell

Once again, Swing provides a ridiculously easy way to do this, using the `JOptionPane` class (which we used earlier to display the "Save Changes" dialog box). Here, we want to use the `showMessageDialog` method, as shown in **Listing 14**.

Listing 14: Displaying the application About box

```

    public void about () {
        ImageIcon penguinIcon = new ImageIcon
            (JaVeez.class.getResource("penguin.jpeg"));

        JOptionPane.showMessageDialog((Component)null,
            "A QuickTime movie player application"
            + "\nbuilt with QuickTime for Java. \n \n"
            + "\u00A92003 by Tim Monroe",
            "About JaVeez",
            JOptionPane.INFORMATION_MESSAGE,
            penguinIcon);
    }
}

```

When we call the `about` method, `showMessageDialog` displays the dialog box shown in **Figure 7**. This is remarkably close to the target About box shown in **Figure 6**.

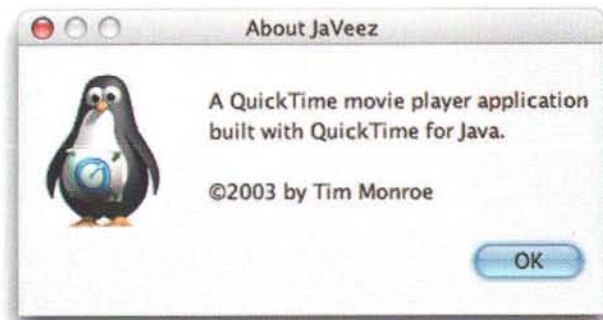


Figure 7: The About box of JaVeez

The parameters passed to `showMessageDialog` should be fairly obvious. The first parameter is the parent frame; in this

case we pass null so that the message pane is embedded in a default frame that is centered on the main screen. The second parameter is the message string; notice that we can embed the newline escape character "\n" into the message string, as well as the escape sequence "\u00A9" (which is the Unicode value of the copyright symbol "©"). The third parameter is the title of the enclosing frame. The fourth parameter is ignored since we are passing an icon as the fifth parameter. Here we are using the standard penguin image, which we added to our project. We need to call the `getResource` method to load that image from the application bundle.

CONCLUSION

In this article, we've seen how to extend the basic Java application we built last time to handle movie editing and document-related operations. The `Movie` and `MovieController` classes in QuickTime for Java provide nearly all the methods we need to handle these tasks. And the `Application` class added in J2SE 1.4.1 for Mac OS X makes it particularly easy to open files dropped onto the application icon, display an About box, and handle application quitting.

JaVeez is now pretty much feature-complete. You may be puzzled, however, that we've focused almost entirely on building an application tailored for Mac OS X. After all, a large part of the allure of using Java to build applications and applets is their ability to run unchanged on multiple platforms. (The Java programmer's mantra is of course "write once, run anywhere".) So what about Windows? Will JaVeez run unchanged on Windows computers that have QuickTime installed? In a word: no. But the changes required to get JaVeez to run on Windows are indeed quite minimal. We'll take a look at those changes, and perhaps also at a few simple enhancements to JaVeez, in the next article.

ACKNOWLEDGEMENTS

Thanks are due once again to Anant Sonone, Tom Maremaa, Chris Adamson, and Daniel H. Steinberg for reviewing this article and providing some helpful comments.

MacTech®
M A G A Z I N E

Get MacTech delivered to your door at a price **FAR BELOW**
the newsstand price. And, it's **RISK FREE!**

Subscribe Today!
www.mactech.com

Long Distance

3.9¢ Per Minute!

Straight 6 second billing increments

**Excellent rates on intrastate, intralata/toll calls
and international calling with no term contract.**

**Toll Free (800/888/877/866) service,
same low per minute rate for incoming calls.**

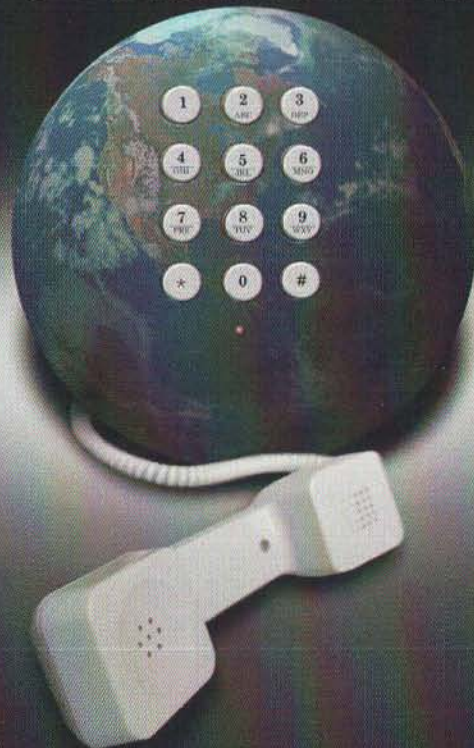
10 cents per minute calling card.

Detailed billing directly from Capsule Communications, a Covista Company.

Quality electronic and telephone customer support.

**No monthly billing fee if you sign up for AUTOPAY billing
option or if your bill is over \$20.00 each month.**

(NOTE: \$1.00 billing fee is charged when your bill is under \$20.00 for all non-Autopay customers.)



www.lowcostdialing.com

By Guyren G. Howe

REALbasic Review

A better Visual Basic than Visual Basic

I always figured that my plans for world domination hinged on being able to quickly write great, easily maintained programs on all the major platforms. I also need my less experienced minions to easily learn to do the same.

Quake in fear, for REALbasic may be just what I'm looking for.

In this review, I'm going to give you a detailed tour of REALbasic, pointing out the interesting features of the language, the IDE, the debugger, the company and the community (because the latter are also very important to the value of a programming language). My aim will be that by the end, whatever your needs, you'll have a good idea whether REALbasic will help you to fill them.

I'm going to mix up talking about the existing 5.2 release and what looks likely to make it into the 5.5 release. I'm doing this because REALbasic is developed at a snappy pace — a new version every six months — and because some of the impending features might well be important to you. Contrasting the two releases will also give you an idea of how quickly the product is improving.

BEGINNER FRIENDLY

Some of what follows will be a bit technical, so I wanted to put this right up front: REALbasic is excellent for beginners. If you're a beginning programmer just wanting to knock together some "basic" programs, REALbasic has that covered. If you're looking to learn programming, REALbasic is very accessible, and you can gradually ramp up to using more sophisticated techniques once you're ready for them.

If you are interested in using REALbasic as a vehicle for learning to program, or if you have never taken a full computer science course and you'd like to improve your programming skills, check out the *Curriculum Project* that I wrote for REALbasic. It's a good basic computer science curriculum in thirty lessons, available from the REALbasic website (www.realbasic.com).

Documentation

A really important thing right out of the box: the REALbasic documentation is actually fairly good, and certainly better than I've seen for many programming tools.

There is a built-in reference that's almost always accurate and sufficient — once you get up to speed, this is mostly what you'll refer to. There are also a *Quick Start* and *Tutorial*, which are fine as far as they go, which isn't very far. There's a *User's Guide* (a quite good, more extensive and useful introduction) and a *Language Reference* (a printable version of the built-in help). There are also a bunch of smaller ReadMe files on such things as database and Internet capabilities. These are concise and accurate, but far too short. Quite a few of these subjects, (and other subjects that for some reason don't even earn a README, such as 3D graphics), could do with extensive guides in their own right.

A FUN DEVELOPMENT ENVIRONMENT

The Integrated Development Environment in REALbasic is quite effective. Not the world's most glamorous, but it strikes a nice balance between simplicity and power. **Figure 1** shows the IDE in action.

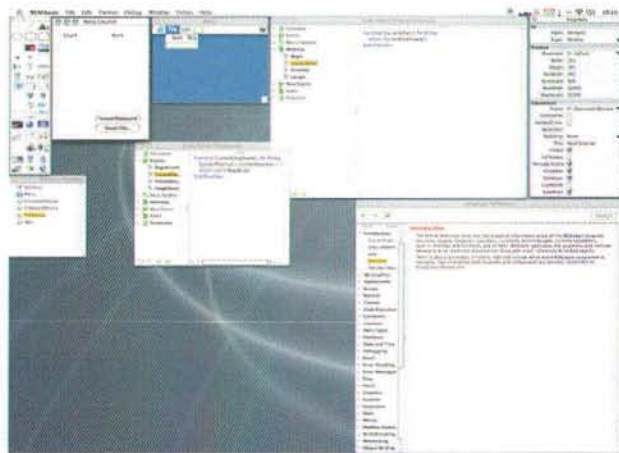


Figure 1: The REALbasic Integrated Development Environment in action

Guyren G. Howe works in artificial intelligence research, after years of work as a technical writer and developer. He is married with one child, is an Australian, and lives in Austin, Texas. Guyren has been working with REALbasic for several years. Most notably, he wrote the REALbasic Curriculum Project, an extensive computer science curriculum, for REAL Software.

This is great at 1600x1200, but it can get a little crowded on my 12" PowerBook's screen. On the other hand, I've never seen a development environment that didn't (okay, other than *Emacs*...).

Most of the action will take place in the window editor and the code editing windows.

A nice code editor

The code editing window isn't the fanciest I've seen, but it gets the job done without fuss (**Figure 2**).



Figure 2: The Code Editor in action

You get one code editing window per class/window/module, combining a simple inspector for all the attributes of that thing on the left, with the associated code editor on the right. The code editor provides comprehensive keyword completion (picking up declarations as they are made, and properly respecting scope), automatic coloring and indentation, along with gobs of little niceties like fairly comprehensive contextual menus, window splitting, and universal drag and drop. I find the relatively spare and uncluttered interface refreshing compared to many others — have you looked at the Visual Basic IDE recently? Ugh.

A fine window editor

REALbasic clearly tries hard to help you create a full-blown user interface. It has a great window editor that uses automatic snapping according to Apple's User Interface Guidelines (snapping to distance between controls, distance from window edges, and to the center lines of the window) to make user interface design almost effortless. **Figure 3** shows REALbasic's comprehensive collection of user interface widgets.



Figure 3: REALbasic's user interface toolkit

The most interesting widgets here are (top to bottom):

- a *Placard* control, specifically for building the little information panel to the left of the horizontal scroll bar in a window (actually quite a nice contribution to making an effective user interface easy to throw together);
- a *BevelButton* control, used for making the kinds of new-fangled buttons all the kids are using these days, combining any or all of text, pictures and pop-up menus;
- a fairly sophisticated spreadsheet-like *ListBox* control;
- a cool 2D animation *SpriteSurface* control;
- an even cooler *Rb3DSpace* control, able to employ a pretty healthy set of OpenGL 3D graphics features;
- a QuickTime movie player;
- Windows OLE container support (for Windows applications, of which more below);
- Mac OS X toolbar and attendant buttons;
- a couple of widgets that take a little of the pain out of doing database interfaces (more below);

- a QuickTime music player; and
- Contextual menus.

As well as the user interface controls, REALbasic lets you fairly easily create OS X *sheets*, *drawers* and even Apple's new, hideous metal windows, along with the full complement of (in many cases, even more hideous) OS 9 window styles.

Oh, and there's a nice menu designer and a very nice event- and class-based way of managing menus, which let you either centralize the menu activation and handling, or distribute the handling around whatever points in the program you feel is appropriate. A good example of the numerous sweet design choices that make REALbasic either very beginner-friendly and straightforward, or surprisingly deep and powerful, depending on how you use it.

All in all, REALbasic couldn't do a whole lot more to help you to create a slick user interface.

Project manager

REAL Software is testing a major new feature in 5.5: the REALbasic Project Manager. This is a simple server to synchronize the work of multiple developers. It provides accounts for multiple developers, and a simple check-out/check-in/synchronize capability for local copies of a project. It's a welcome feature, and I hope they continue to develop it with such features as history and rollbacks. It would also be nice if the check-out/check-in system offered an alternative CVS-style differencing model as an option.

VISUAL BASIC + JAVA OBJECT MODEL + CREATIVE THINKING = GOOD

REALbasic draws heavily on Visual Basic, to the extent that REAL Software supplies a free Visual Basic converter: a project converted from Visual Basic will still need some work before it's functional, but much of the code in a typical Visual Basic application will work exactly the same in REALbasic. I've heard of a few quite complex VB projects converted to REALbasic in just a few days.

The Visual Basic language (Visual Basic was bought, not invented, by Microsoft, by the way) is a generic, friendly, predictable, structured programming language. Having this language at its heart, REALbasic nicely satisfies the *principle of least surprise* — things mostly just work the way it seems they ought to. Here's a bubble-sort algorithm:

A bubble sort in REALbasic

The heart of REALbasic is the Visual Basic language — a nice, generic structured programming language

```
//Sort a list using a BubbleSort.
Dim Done As Boolean
Dim Counter, Temp, Limit As Integer

Limit = UBound(Numbers)
Do
    Done = True
    For Counter = 0 To Limit - 1
        if Numbers(Counter) > Numbers(Counter + 1) Then
            Temp = Numbers(Counter)
            Numbers(Counter) = Numbers(Counter + 1)
            Numbers(Counter + 1) = Temp
        end if
    Next
    Limit = Limit - 1
Loop Until Done
```

```
Done = False
End if
Next
Limit = Limit - 1
Loop Until Done
```

But REALbasic's programming language is far from being a slavish copy of Visual Basic. The language extends Visual Basic into a grown-up object-oriented programming language, mostly by adopting the single-inheritance-plus-interfaces OOP paradigm of the likes of Java.

Major Language Features

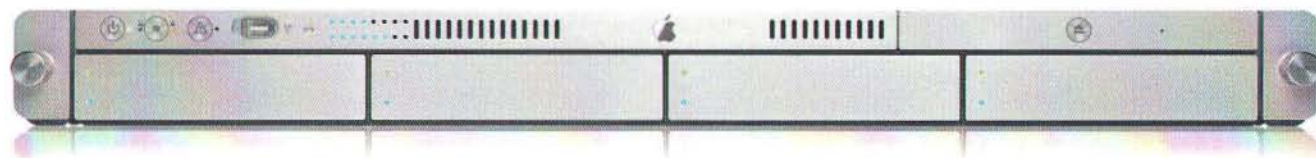
- Traditional method overriding. REALbasic also supports a novel and marvelous class extension mechanism called *Events* (more on that below);
- Single inheritance, with interfaces;
- Method and operator overloading;
- Simple compiler conditionals (so you can compile, say, different code for different platforms. There are no macros, though);
- Robust, typed exceptions with a full *try-catch-finally-end try* block (added in 5.5 — earlier releases only had exception catching at the end of a method);
- Comprehensive safety checking (array out of bounds, nil variable dereference, and so on), with the ability to disable any of the safety checks in particular parts of the code, for speed;
- "Bare" methods and properties (having global scope, and not being part of a class);
- Simple namespaces (in 5.5);
- Public/Protected/Private scope (expanded in 5.5 from Public/Protected) for methods and properties;
- Separate scalars and objects, as in Java (many would consider this a mis-feature) — and no sign of auto-boxing (automatic conversion of scalars to objects and back);
- A complete set of basic math operations, including bit operations, and even quaternions (no complex numbers, though — the quaternions are provided for 3D math);
- The ability to define your own "bindings" for classes, that support connecting objects together in the user interface by just dragging a line between them, much as you can in Apple's Object Builder;
- "Getter" and "Setter" methods invoked through direct property access syntax;
- Latently-typed (good) automatically type-converted (bad) *variants*, just like in Visual Basic.

Conspicuously absent from the list are any real introspection features, serious modularity support, or classes as first-class objects in their own right, with their own methods, properties and events. These are significant omissions from an otherwise robust object-oriented programming language.

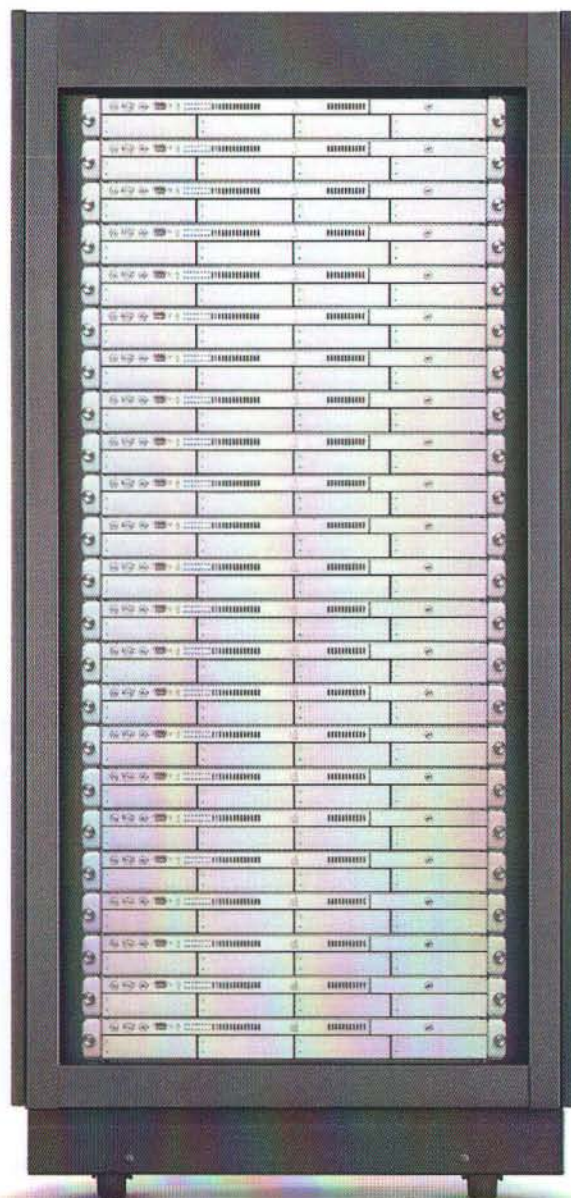
Inheritance done different, and done right

There are one or two ways in which REALbasic's designers

Xserve



Density optimized rack mounted Mac OS X Server



UNIX-based Server Solutions from Apple
Nearly half a terabyte of storage per 1U
630 gigaflops of processing power
Hot-Swappable drives
Industry-standard 1U rack-optimized design

There has never been a better time to buy...
Small Dog Electronics carries
factory refurbished models too
offered at substantial savings!
(subject to availability)



**Small Dog
Electronics**
www.smalldog.com

1673 MAIN STREET, ROUTE 100, WAITSFIELD, VERMONT

 Apple Specialist

To learn more: <http://www.smalldog.com/xserve/>

have been prepared to develop creative new contributions to REALbasic's programming paradigm. One in particular, which goes by the everyday name of *Events*, is a marvelous idea that I've never seen in any other programming language. I'm going to unpack this for you, because it's an important language feature you won't be familiar with, and because I think more programmers should see this idea.

Events in REALbasic provide a top-down class extension model, by which a superclass can declare an API for delegating to its subclasses. Rather than a subclass just overriding its superclass's methods willy-nilly, in REALbasic, the code in the superclass can explicitly issue calls to code declared in its subclasses. This will very often significantly simplify maintenance, because if you need to modify the superclass, you generally know what the subclasses will do as a result. Once you get used to it, explicit communication between parts of the class hierarchy just feels *right*.

Here is an example of an abstract superclass and a concrete subclass, using events. This code implements part of a collection of character streams.



Figure 4: An abstract base class, *CharacterProvider*

The example shows a method calling a *New Event*. New Event declarations take exactly the same form as interface declarations, which is more or less what they are. A *New Event* specifies a sort of internal API that code in subclasses can provide hooks for. Code in the parent class can then invoke any of its New Events in exactly the same way that it invokes a method. The parent class doesn't need to know any details of what the subclass code does in response to the event, or where, how or even *if* the event is handled; it just "delegates down". In the example superclass, all of the methods are nothing more than one-liners that immediately invoke the corresponding event. This is the standard way to implement an abstract base class in REALbasic.

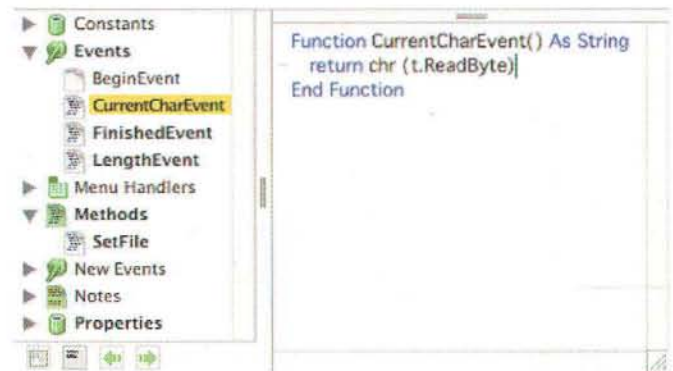


Figure 5: A subclass, using Events

Figure 5 shows a concrete subclass of the previous class that provides its characters from a text file. The screen shot shows code in one of the event handlers. Most of the code in this class is in its *event handlers*, but it also adds an extra method to set the file the instance of this class will read characters from.

It is possible for a chain of subclasses to contribute to the handling of an event. It is not required that a subclass provide a handler for an event it inherits. Rather, the *most immediate subclass* that declares a handler for an event 'swallows' the event, blocking 'lower' subclasses from seeing it. But the subclass with the event handler can, in turn, re-declare a New Event with the same name and arguments, and delegate further opportunities for behavior down the line. It can also define and call entirely new events, thus presenting a richer or modified set of behaviors for subclasses to extend further.

Events can be either method-style or functional (returning a value). A call to an unhandled method-style event is just ignored; calling an unhandled functional event returns 0, an empty string, *Nil* or the like.

The built-in REALbasic classes use this paradigm for extension, and it is quite easy to implement, say, a subclass of the built-in text editing class that only lets the user enter numbers, but otherwise works the same, including invoking the same event handlers on any further subclass, in case such a subclass needs to modify this behavior further.

It's a demonstration of the clarity events bring to inheritance that one can almost always just take one event-based inheritance chain, "paste it" at the bottom of another, and it just works. It almost seems as though this ought to make possible a form of event-based multiple inheritance in REALbasic.

One of the built-in classes is the *Canvas* control. This is essentially an empty control, providing a graphics space into which a subclass can draw, and providing every conceivable kind of user interface event for the control to respond to. It is fun to take this blank slate and implement an interesting user interface control with its own display capabilities, events and methods for further subclasses to draw on.

The event-based inheritance paradigm is, in short, a delight.

Plug-in authoring and the C interface

It is possible to drag classes, class hierarchies and modules from REALbasic out to the file system and into another project, including doing so as a soft “link” to the file (so the project automatically re-imports the file on launch, and any changes to the class are written out to the external file — great for classes shared between projects). These exports can be encrypted if you wish to make them available without revealing the source code. You can also choose which of their properties can be set in the IDE, if an instance of the class is dragged into a window.

However, multiple classes — even in the same hierarchy — have to be exported as separate files, and have to be dragged into another project in a superclasses-before-subclasses order, or things break. This is just stupid, has been there since day one, and shows no sign of being fixed soon.

I would really like an ability to bundle up a class hierarchy and export it as one object. And it's a pity I can't set a project to automatically keep its entire contents as separate files, so I could use, say, CVS to manage the project (but see the discussion of the *Project Manager*, below).

REALbasic also provides a decent API for writing your own plug-ins in C. Unfortunately, the documentation for this hasn't kept up to date with changes to REALbasic. The documentation discusses making old-style resource-based plug-in using CodeWarrior on a Mac. The only deference to new features that I could see was a change history, and a brief mention in the README that old-format plug-ins should be converted to the current format using a supplied utility.

I understand that it is possible to author Windows plug-ins using Visual Studio, but this is essentially undocumented. You can work out how to do it by looking through the mailing list archives and asking the right questions, but the documentation here really needs to be fixed.

RBScript

Another relatively unusual language feature: REAL Software developed the new compiler they introduced in 5.0 by first developing “RBScript”, which is essentially the REALbasic language compiler, without all the hooks into the IDE and the toolkit. This has been developed into a built-in, compiled scripting language. The language is by default completely sandboxed — it can't access anything outside itself. The developer can attach one object to the RBScript before compiling and running it, and the methods and properties of the object are available as globals in the script. So you can un-sandbox any capability you wish, and you can provide selective access to the rest of your application quite easily. Since it is also possible for the application to “covertly” put extra code at the top of a user-supplied script, providing an arbitrary number of ready-made classes and other features, it is entirely possible to create an arbitrarily full-featured scripting environment for a compiled REALbasic application, with comparatively little effort. This is a great feature, and I'm itching to try a project where I put a lot of, say, business logic into RBScript — I could even

keep the code in a database, or make it downloadable. Used effectively, RBScript could make a widely-deployed application unusually easy to support, upgrade and customize.

Debugging and Profiling

REALbasic's debugger is decent, but nothing amazing. It offers the standard click-to-put-a-breakpoint in the left margin of the code editor, but no easy way to find or temporarily disable the breakpoints you have. No conditional breakpoints (although in 5.5, you can issue a `Break` command to activate the debugger, so you can do whatever you like with that).

Once you fire it up, the debugger is straightforward and effective (**Figure 6**).

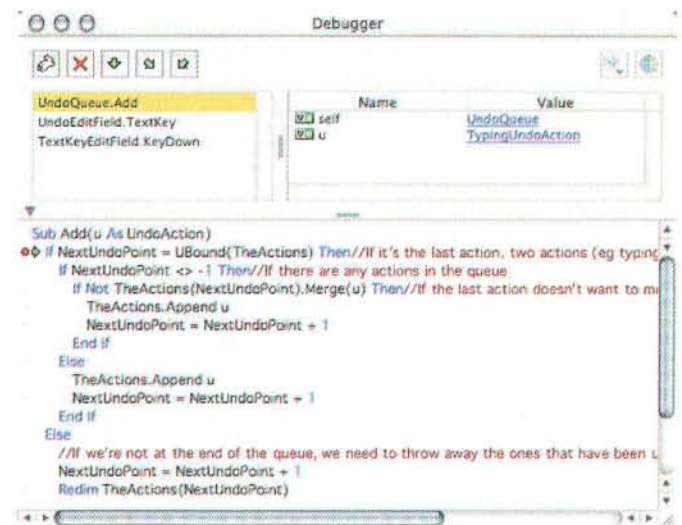


Figure 6: The REALbasic Debugger

The debugger covers the basics: you can see the stack, your local variables, and globals. You can step over/into/out. A nice touch is that you can set the debugger to automatically activate when any exception is raised.

When you click on a variable in the locals pane on the right side of the debugger, you get a nice little inspector (**Figure 7**), and by clicking on object references (to open a new inspector on that object in a new window), you can drill down through a data structure as far as you'd like. A little difficult to see the whole data structure that way, but not bad. You can also modify scalar values while the program is running.

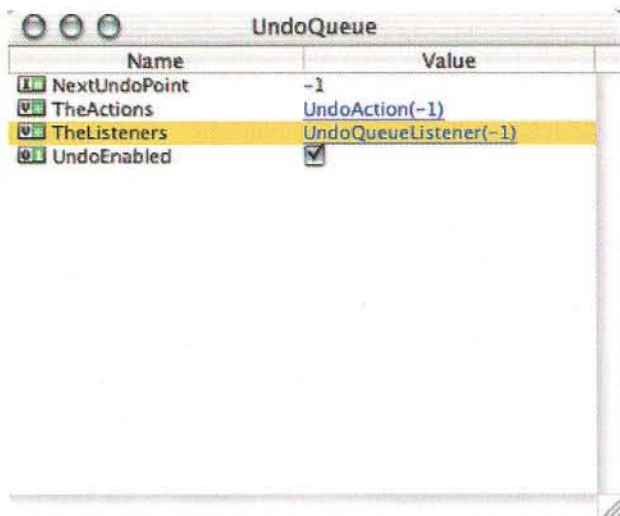


Figure 7: An Object Inspector

Remote debugging

For 5.5, REAL Software is testing remote debugging. This makes it possible to debug a program running on one computer, from another computer, even running the debugger on a different platform to the one the program itself is running on, or running the debugger remotely, via the Internet. This is a marvelous feature, and I hope it makes the cut for 5.5.

No profiling

Don't look for automated profiling, because there isn't any. It's understandable, with the focus and style of REALbasic, but I think one of the best features a language can have is a good profiler, and I hope REAL looks into this before too long.

No Project Browsing

Seriously missing in action is any means of browsing the class hierarchy. I find it hard to believe that I can't open a window and see a tree of the classes and subclasses in the program. Finding where you are can be a significant problem in a complex program.

5.5 adds a contextual menu choice that takes you from a method call to its definition, so at least there is *a little* progress on this front.

THE TOOLKIT

While REALbasic's "batteries included" don't approach those you'll find in, say, Python, you will find a very respectable and rapidly expanding collection of ready-to-go features "in the box". I'll discuss a few of the most important features below, but here is a quick list of some other major features:

- A fairly extensive set of Apple Event sending and receiving capabilities;
- Extensive vector drawing primitives (including Bezier curves, rotated objects and text, grouping, and employing bitmapped

images as part of a vector drawing);

- Fairly extensive bitmap manipulation primitives (through either Quartz — pretty! or QuickDraw — fast!);
- QuickTime editing of movies in code (surprisingly complete, including access to QuickTime effects);
- Complete file system access, including random-access read-write files, sequential access to files, getting and setting file and folder properties, and access to Resource Forks;
- Full access to the keychain and the clipboard (apart, oddly, from being able to find what arbitrary types of information are on the clipboard);
- A fast, raw *MemoryBlock* object type;
- A 3D sound class, with quite rich capabilities drawn from QuickTime;
- Access to the Registry (on Windows);
- Sending and receiving *Apple Events*;
- Straightforward multi-threading (sadly, only on one processor), with semaphores and critical sections;
- Ability to control a shell, on both OS X and Windows (strangely, though, no support for *StdIn*, *StdOut*, or *StdErr*);
- Comprehensive POP/SMTP Email sending and receiving, including with SSL (in the Professional release);
- HTTP (with SSL, in the Professional version) protocol support for sending and receiving;
- Regular Expressions (but not as part of the language — this is a set of classes you have to instantiate, set properties on, and invoke — so not remotely as convenient as in, say, PERL);
- Fairly thorough support (I understand — my language skills are limited to the Roman alphabet) for text encoding and Unicode.

DATABASE CAPABILITIES

REALbasic's database capabilities are quite decent on the whole, although some gaps in the "batteries included" are disappointing. Also see *Printing*, below, for a more serious obstacle to database applications.

REALbasic includes plug-ins giving support for a nice range of common databases:

- ODBC;
- PostgreSQL;
- MySQL;
- 4th Dimension Server;
- OpenBase;
- FrontBase;
- Sybase
- Comma-separated files;
- .dbf (dBase) files;
- dtd;
- Oracle.

REALbasic also includes a plug-in for a competent basic single-user proprietary database format.

MacTech[®]

M A G A Z I N E

The source for developers and the technical market since 1984.

CRASH PROOF

- No installer needed
- No download times
- Readable in your favorite reading locations
- No drain on your system's resources
- The best reader user interface mechanism
- Available Monthly
- Delivered painlessly to your doorstep.

*Authored by a number of industry experts
working in the real world*

Toll Free: 877-MACTECH

Get it today RISK FREE at <http://www.mactech.com>

For databases not supporting SQL directly (most notably, the built-in database and 4th Dimension), REALbasic provides an interpreter for a useful-but-small subset of the SQL language.

There is also a very good-looking third-party database plug-in available called *Valentina* that seems to be getting a lot of attention in the REALbasic community.

All of the database connections share their more important methods and properties and have a common base class, so to the extent that your SQL code is portable between databases, your REALbasic application can be switched from one type of database to another quite trivially.

Access to the data is through a standard cursor-type object. Some standard special-purpose interface controls and some special features of some of the general-purpose controls (such as the EditField and the ListBox) make it quite easy to put together a basic interface to scroll through a list of records, double click a row and go to a detailed view of a record, update fields, click through records one at a time... The automated features don't get in the way of doing some of the work yourself should you need something out of the ordinary. Also, insofar as you can do your data manipulation through the one-record-at-a-time cursor, you can make your code even more independent of the particular underlying database.

I wouldn't say that a REALbasic database user interface is remotely as quick to set up as, say, a FileMaker interface. But it's a couple of orders of magnitude more flexible without being anything like that much more work.

On the other hand, if you want to develop a GUI database application of any complexity, you'll probably need some user-friendly query, import/export or customizable reporting applications, and REALbasic doesn't come with anything like that. REALbasic provides everything you need to write those from scratch, but doing a good job of these kinds of applications would be a serious effort.

Plug-ins may help here. I've heard nice things about *dbReports* (<http://www.abdatatools.com/dbIndex.html>), for user-customizable reports. There may well be similar alternatives for the import/export and querying, but after spending some time googling for them, I came up short. More on plug-in availability below.

Sadly, REALbasic doesn't take the opportunity to provide an object database, or anything else to help bridge the object-relational divide.

PRINTING

Far and away, the biggest problem you'd have doing a significant database application (or, indeed, many kinds of programs) in REALbasic is printing.

REALbasic provides only very low-level, QuickDraw-like printing capabilities. To print from a REALbasic application, you get a special graphics object into which your code draws, then you print it when you're done.

The graphics drawing features of the REALbasic language provide good basic capabilities for this — you can ask

REALbasic to draw a string or styled text into a box of a certain width, and it then tells you what height it was, for example. But doing any kind of complex report in code like this is intractable for the upwards of hundreds of reports you might have in a typical bespoke database application.

It's a great shame, too, because REALbasic would otherwise be excellent for such development. This is far and away the most serious hole in REALbasic's feature set.

I understand that REAL Software is working on a new printing capability, but it's not looking like anything is going to appear in 5.5, at time of writing.

MICROSOFT OFFICE PLUG-INS

Relatively unknown fact: Microsoft's Mac development team used REALbasic to prototype the user interface for the last version of Internet Explorer for the Mac, and they continue to use REALbasic, even publishing at least one part of the last version of Office written in REALbasic (below).

It also appears as though Microsoft are working with REAL Software on a Microsoft Office automation plug-in for REALbasic. The feature isn't *absolutely* complete yet (particularly lacking the ability to trap events in Office applications), but it can probably do anything you're really likely to need — the examples for the Plug-ins create and modify a Word document, create a PowerPoint presentation, import a picture into PowerPoint, and create an Excel spreadsheet. The features already available can do most of what you are likely to need (automated creation of pretty reports, slideshows and spreadsheets from REALbasic).

The code is also fairly straightforward.

```
REALbasic code to create a Word document
It is fairly straightforward to use the Office Automation
plug-in.
Sub Action()
    dim word as wordapplication
    dim doc as worddocument
    dim style as wordstyle
    dim v as variant

    word = new wordapplication
    doc = word.documents.add
    doc.range.text = editfield1.text

    v = doc.paragraphs.item(1).style
    if v.objectvalue isa wordstyle then
        style = wordstyle(v.objectvalue)
        styletext.text = style.namelocal
        stylefontname.text = style.font.name
        stylefontsize.text = str(style.font.size)
    end if

    exception err as wordexception
    msgbox err.message
End Sub
```

One very nice feature is that the same code should work on both Macintosh and Windows (modulo some features that, at time of writing, had not been implemented on the Mac yet).

Sadly, as with a few of the most interesting features of REALbasic, the documentation is woefully incomplete. A terse README or two and a few very simple examples constitute all

the documentation I'm aware of. I think the REAL Software folks are expecting the developer to look to other *Visual Basic for Applications* documentation to fill in the blanks here (the plug-ins exploit REALbasic's similarity to Visual Basic to re-implement virtually the same API). But it would be nice if, at least, they pointed the user to a good source for this information.

Note that you can also exercise extensive control over a huge range of Macintosh and Windows applications using Apple Events, AppleScript (below) or OLE.

APPLESCRIPT

A *great* feature is that you can compile an AppleScript method, drag the AppleScript file into REALbasic, and just call it from your REALbasic code, no other work required. This is a heck of an integration feature, a very quick and easy way of talking to all sorts of applications, and a very nice way of putting a user interface on an AppleScript.

NATIVE TOOLBOX CALLS

On both Macintosh and Windows, it is possible to access toolbox calls. Enough low-level bit-wrangling features are available (including being able to do things like get handles for REALbasic windows) that you can access most toolbox features if you know what you're doing. And if you don't, this feature is the basis of a lot of cheap-or-free classes providing access to most OS features that aren't already covered by REALbasic.

On Windows, you can also call into DLLs, and in 5.5, REAL is testing callbacks into REALbasic methods for both Macintosh and Windows.

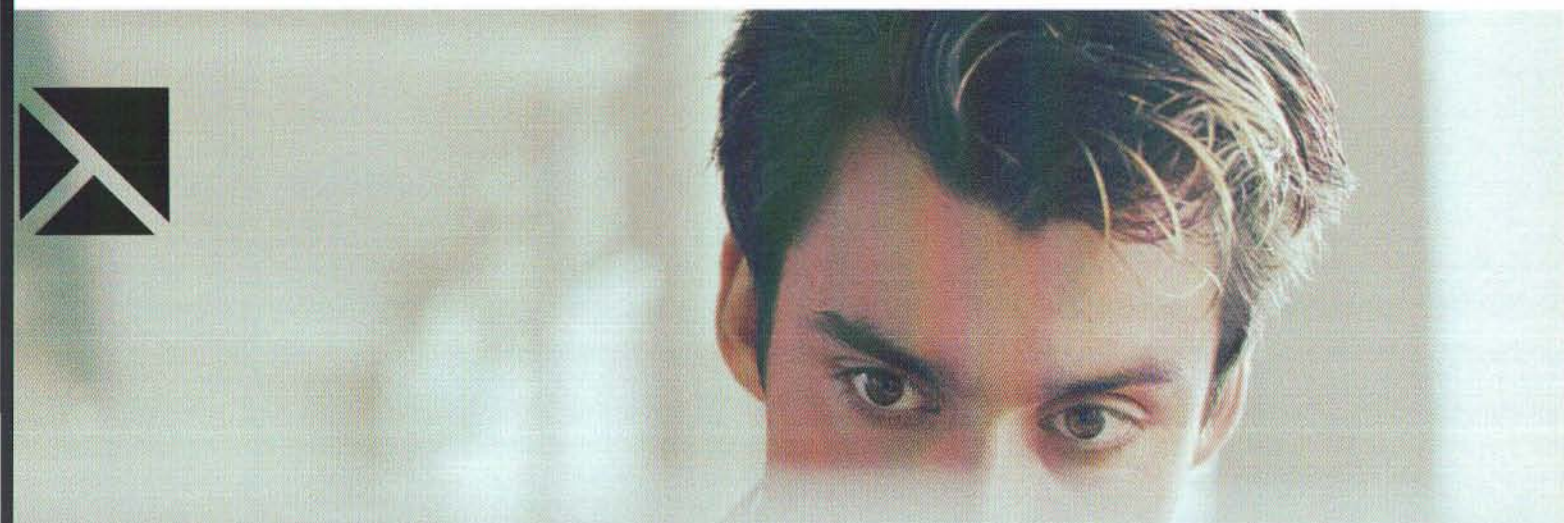
CROSS-PLATFORM DEVELOPMENT

REALbasic is becoming one of the best ways around to write cross-platform code. There is a functionally equivalent IDE for Windows, and you can compile to Windows on the Mac or vice-versa (with the Professional version). The integration with Windows isn't as complete as is the integration on the Mac, but it's pretty decent. In particular, the most important piece, being able to use OLE plug-ins, is there.

For 5.5, REAL Software is beta-testing a Linux compile. In developing their Windows version, REAL implemented compiling for Windows on the Mac in early releases, following in later versions with the Windows IDE. This is almost certainly what they're doing for the Linux transition, which targets GTK. Once they get the kinks out of the Linux compile, REALbasic will have a good claim to being the best cross-platform GUI development story around.

THE COMPANY

REAL Software is a pretty pleasant company to deal with. They have a couple of very active developer email lists (one for the regular release, one for the betas), as well as a few other lists for the likes of beginners and game developers. Virtually all of



End the compromise with perfect quality and file size.

Produce highly compressed, lossless QuickTime videos for delivering screen capture training videos, animated text and graphics.

Try it FREE today at www.techsmith.com

TechSmith
ENSHARPEN™
Video Codec

the REAL Software staff post on the email lists from time to time, and many of them post just about daily, including the president of the company, and all the developers.

REAL Software is very open with its development process, putting regular development releases out for general consumption, testing and debate. And they're very good-humored about the inevitable problem with such a development strategy: people complaining, sometimes bitterly, about bugs in alpha and beta software.

They also have an excellent bug tracking and feature requesting system, in the form of a web-accessible database. You can add to it only after first searching for an existing report, and you can add comments to an existing report or "vote" on it, adding your weight to the need to address the report. And REAL does seem to try fairly hard to address these things in order of popularity.

There is a developer's program offering several tiers of membership, providing such things as expedited support.

THE COMMUNITY

REALbasic boasts a helpful and robust community, focused on the mailing lists. There are also a great many web sites concerned with REALbasic, several books, and quite a few developers sharing all manner of code, tutorials, plug-ins and other tools.

On the other hand, there is no CPAN-type code library for REALbasic. Partly, this is the language design that doesn't (yet) provide the sorts of features you'd need to really easily share code (I'm thinking particularly of a real modules and namespaces feature — some of the new features in 5.5 are a good step in this direction). But none of that is a show-stopper — it could still be easier to find plug-ins, for example.

There is one site, rbgarage.com, that is clearly aspiring to be a CPAN for REALbasic. It's a nice site, clearly someone is putting in some effort here. It remains to be seen whether they will get the momentum going to become the kind of canonical point-of-origin that CPAN is, but I wish them well.

THE PROOF IN THE PUDDING

I always like to find out what *real* projects have been written with a programming language. Without further ado:

- *Art Directors Toolkit*, a designer's toolkit, which Apple bundles with all PowerMac G4 Towers and Powerbook G4 Laptops.
- *Microsoft Query for Mac OS X*, which uses Open Database Connectivity (ODBC) to allow you to import data from databases into Microsoft Excel X.
- *Auto-Illustrator*, an "experimental, semi-autonomous, generative software artwork and a fully functional vector graphic design application to sit alongside your existing professional graphic design utilities".
- *Whistle Blower*, a server monitor and functional testing utility.

- *Intellimerge*, e-mail merge software for Mac OS X and Mac OS 9.
- *Coldstone Gaming Engine*, an easy to use game development tool that allows you to create professional stand-alone games that run on Mac OS, Mac OS X, and Microsoft Windows.
- *A-OK: Wings of Mercury*, a realistic simulation of the Mercury spacecraft.
- *SpamFire*, An anti-spam email filter.
- *MediaEdit*, a movie and graphic editing tool.

WHAT YOU GET

REALbasic comes in two versions (Standard and Professional) for two platforms (Macintosh — Native and Classic, and Windows).

The extras in the Professional version are:

- Cross-platform compiles;
- The database features;
- SSL for the Internet features; and
- a better "server" version of networking sockets.

In the US, at time of writing, a Standard version can be had for \$99.95, and Professional for \$399.95. That's for the download-only version. A little more gets you over 1000 pages of dead tree documentation and a CD. There is also a somewhat more expensive version that gets you a year of upgrades.

Upgrades from the previous version are \$79.95 (Standard) and \$169.95 (Professional). If you're serious about REALbasic, the subscription is a non-trivial saving on the upgrade.

There are also some Academic pricing options, volume purchase options, and the like.

CONCLUSION

You could call high-level languages an obsession of mine. I've tried many, many languages. Compared to REALbasic, I've tried Deeper languages (Scheme, Dylan), more broadly accepted languages (C++, Python, PERL, ...), languages that are easier to use (Python), and languages with better IDEs (Java with Eclipse or Objective C with Project Builder).

But I've not used *any* language or IDE that makes good GUIs easier to construct than does REALbasic. That would be a great argument for REALbasic on its own, but the clincher for me is that REALbasic scores a consistent "really, pretty decent" on depth, acceptance, ease of use, and the IDE — something I wouldn't say for any other language I know.

REALbasic is an effective tool for either beginners or experts and it's fun to work with. As long as its one really major limitation (printing) isn't a problem for you, it's a good candidate for a very broad range of development projects.



Peachpit

Essential books for the creative community

Start the New Year Right!

Achieve your new year's resolutions with help from these great guides covering everything from wireless networking to golf to building a business on eBay. It's all here!

>> **The Wireless Networking Starter Kit, Second Edition**

By Adam Engst and Glenn Fleishman
0-321-22468-X • \$29.99 • 560 pages

>> **TechTV's Mod Mania with Yoshi: A Guide to Customizing Your Computer and Other Digital Devices**

By Joshua "Yoshi" DeHerrera
0-7357-1405-3 • \$15.99 • 176 pages

>> **Macromedia Flash MX Professional 2004 for Server Geeks**

By Nate Weiss
0-7357-1382-0 • \$45.00 • 624 pages

>> **TechTV's Guide to the Golf Revolution: How Technology is Driving the Game**

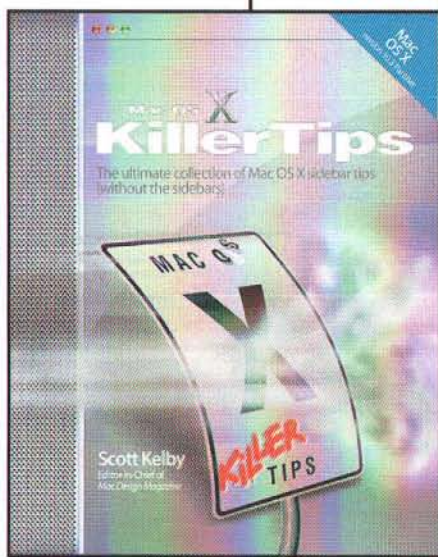
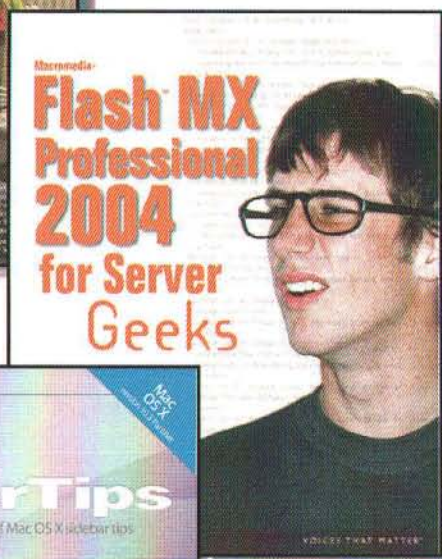
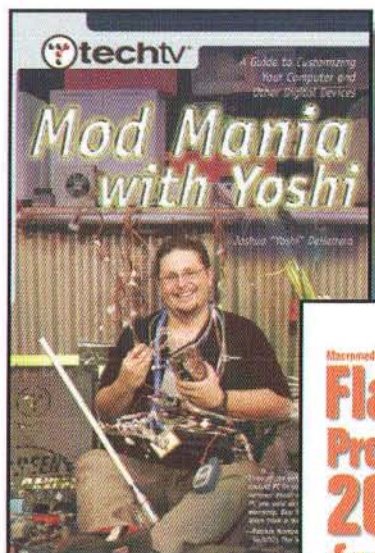
By Andy Brumer
0-7357-1406-1 • \$19.99 • 176 pages

>> **Sell it on eBay: TechTV's Guide to Successful Online Auctions**

By Jim Heid and Toby Malina
0-321-22376-4 • \$24.99 • 176 pages

>> **Mac OS X Panther Killer Tips**

By Scott Kelby
0-7357-1393-6 • \$29.99 • 275 pages



Save up to 30%! Become a Peachpit Club Member today!

Enjoy 10% off all books every day at peachpit.com, earn an additional 10% discount as a Peachpit Club Member, and save 10% on top of that with this one-time coupon! Simply go to www.peachpit.com/mactech0104 and enter coupon code EM-J4AA-MTMF at checkout. It's that easy!



Peachpit Press

**New
Riders**

List of Advertisers

Aladdin Knowledge Systems, Inc.....	21
Aladdin Systems, Inc.....	47
Ambrosia Software.....	5
Asanté Technologies, Inc.....	37
Big Nerd Ranch, Inc.....	13
Brad Sniderman	45
DevDepot	28-29
Electric Butterfly	51
FairCom Corporation.....	1
Fetch Softworks.....	65
Full Spectrum Software, Inc.....	23
Lemke Software GmbH	7
MacDirectory	39
MacTech Magazine	75
MYOB US, Inc.....	61
Netopia, Inc.....	IFC
Paradigma Software	27
Peachpit Press	79
Pearson Education Communications	49
piDog Software	11
PowerGlot Software	33
PR Newswire.....	63
PrimeBase (SNAP Innovation)	25
Runtime Revolution Limited.....	BC
Seapine Software, Inc.	55
Small Dog Electronics.....	71
Sophos, Inc.	9
TechSmith Corporation	77
ThinkFree Corporation	43
Utilities4Less.com.....	67
VVI	17
WIBU-SYSTEMS AG	IBC

List of Products

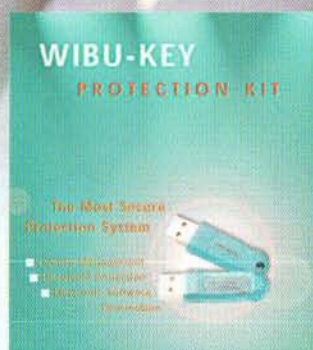
Adobe Press • Peachpit Press	79
Big Nerd Ranch • Big Nerd Ranch, Inc.	13
c-tree Plus • FairCom Corporation	1
Developers Library • Pearson Education Communications	49
Development & Testing • Full Spectrum Software, Inc.....	23
Digital Rights Management • Aladdin Knowledge Systems, Inc.....	21
Ensharpen • TechSmith Corporation	77
Fetch • Fetch Softworks	65
Graphic Converter • Lemke Software GmbH.....	7
HelpLogic • Electric Butterfly.....	51
InstallerMaker, StuffIt • Aladdin Systems, Inc.	47
IntraCore Switches • Asanté Technologies, Inc.	37
Law Offices • Brad Sniderman	45
Long Distance Phone Service • Utilities4Less.com	67
MacDirectory • MacDirectory.....	39
MacTech Magazine Subscription • MacTech Magazine.....	75
Maximizing Your Mac! • DevDepot.....	28-29
New Product • MYOB US, Inc.	61
piDog Utilities • piDog Software	11
PowerGlot • PowerGlot Software	33
PR Newswire • PR Newswire	63
PrimeBase • PrimeBase (SNAP Innovation)	25
Revolution 2.1 • Runtime Revolution Limited.....	BC
SmallDog.com • Small Dog Electronics.....	71
Snapz Pro X 2.0 • Ambrosia Software	5
Software Protection • WIBU-SYSTEMS AG	IBC
Sophos Anti-Virus • Sophos, Inc.	9
TestTrack Pro • Seapine Software, Inc.....	55
ThinkFree • ThinkFree Corporation	43
Timbuktu & netOctopus • Netopia, Inc.....	IFC
Valentina • Paradigma Software.....	27
Visual-Report Tool Developer • VVI.....	17

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

WIBU-KEY Software Protection

Be sure to order a **WIBU-KEY Protection Kit** and find out why small software companies to Fortune 500 enterprises are switching to the WIBU-KEY Security, not **Obscurity** model for all of their software licensing needs, including:

- Common API across all platforms and hardware
- Hardware-based code and data encryption
- Field-upgradable and reusable hardware
- The most cost-effective network licensing solution available
- The only system to employ RID/RED and AXAN security
- Engineered and manufactured to exacting ISO9001 standards



Test the
WIBU-KEY
Protection Kit
sales@griftech.com

Call 800-986-6578

The Key is in Your Hands!

WIBU
SYSTEMS

WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
Email: info@wibu.com

www.griftech.com
www.wibu.com

Protection Kits also available at:

Belgium wibu@impakt.be, Denmark lean@danbit.dk, Finland linebyte@finebyte.com, France info@neol.fr, Hungary info@mrsoft.hu, Japan info@suncaria.co.jp, Jordan, Lebanon starsoft@cyberia.net.lb, Korea dhkimm@wibu.co.kr, Luxembourg wibu@impakt.be, Netherlands wibu@impakt.be, Portugal dubite@dubit.pt, Thailand preecha@dpf.co.th, United Kingdom info@codework.com, USA sales@griftech.com

It'll Give You A Life.

O'ahu, Hawaii- Sun, sand and soft breezes make this one of the most relaxing vacation spots in the world. Don't forget the cool, fruity drink!

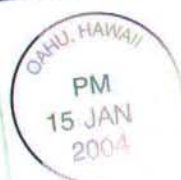
Dear Revolution 2.1

You just keep on
getting better.
Wish you were here.

Missing you terribly,

Love,

Cecil



Revolution 2.1
C/o Cecil's Computer

www.runrev.com

24-7 Relief From Aggravation
(formerly DullAndDreary Coders)
The Corporate World, #1-EASY

But A Geek Is Always A Geek.

**Revolution 2.1: the English like language
designed around the way you think.**

Develop and deliver on 14 platforms, including Mac OS X, Windows and Linux.
Now with support for XML, additional SQL databases, video capture, Unicode,
Reports, enhanced faceless CGI applications, and more.

And now from Dan Shafer, the first "how to" book on Revolution.
Get a headstart with "Revolution: Software at the Speed of Thought",
volume one now shipping from www.runrev.com/revpress.

Thousands of developers have already joined the Revolution. Can you afford to wait?
Pricing starts at \$149. Don't let the revolution in coding start without you.

User Centric Development Tools



Runtime Revolution • 91 Hanover Street • Edinburgh EH2 1DJ • UK
Phone +44 (0)131 718 4333 • Fax +44 (0) 845 4588487 • www.runrev.com • Email info@runrev.com

**Revolution
Studio**



**winner
macworld
eddys**